

## Objective

This example demonstrates how to use the Serial Memory Interface (SMIF) Component in execute-in-place (XIP) mode with external F-RAM™ on PSoC® 6 MCU devices.

## Overview

This example uses the SMIF Component in XIP mode to manipulate an array in an external FRAM device. The workspace contains two projects: FRAM\_XIP and FRAM\_XIP\_DMA. The FRAM\_XIP project uses the SMIF Component and SMIF API to carry out operations. The FRAM\_XIP\_DMA project uses the SMIF Component with DMA to perform transfers to and from F-RAM.

## Requirements

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

**Programming Language:** C (Arm® GCC 5.4.1)

**Associated Parts:** All PSoC 6 MCU parts

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

## Hardware Setup

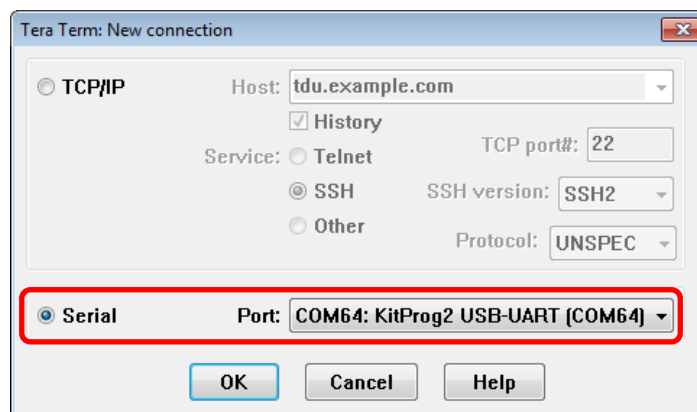
This example uses an external F-RAM device compatible with the CY8CKIT-062 and CY8CKIT-062 WIFI-BT kit. For either kit, ensure that footprint labeled U5 is populated with the Excelon™-Ultra (QSPI F-RAM).

## Software Setup

This section describes how to set up a serial (UART) connection using Tera Term on a PC to communicate with the PSoC 6 BLE Pioneer Kit. Tera Term is a free software terminal emulator for Windows, which can be downloaded [here](#). Other software terminal emulator programs such as PuTTY can also be used.

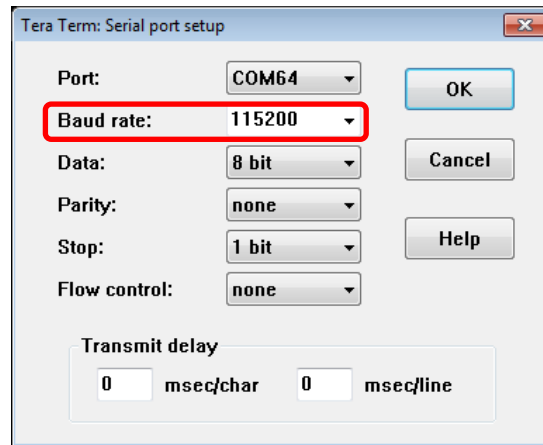
1. Connect the PSoC 6 BLE Pioneer Kit to the PC using the USB cable.
2. After installing Tera Term, open the program and select the KitProg2 device in the Port drop down menu. Click **OK**.

Figure 1. Tera Term Port Selection



3. In Tera Term, select **Setup > Serial port** and set **Baud rate: 115200, Data: 8 bit, Parity: none, Stop 1 bit, Flow control: none**. Click **OK**.

Figure 2. Serial Port Configuration Settings

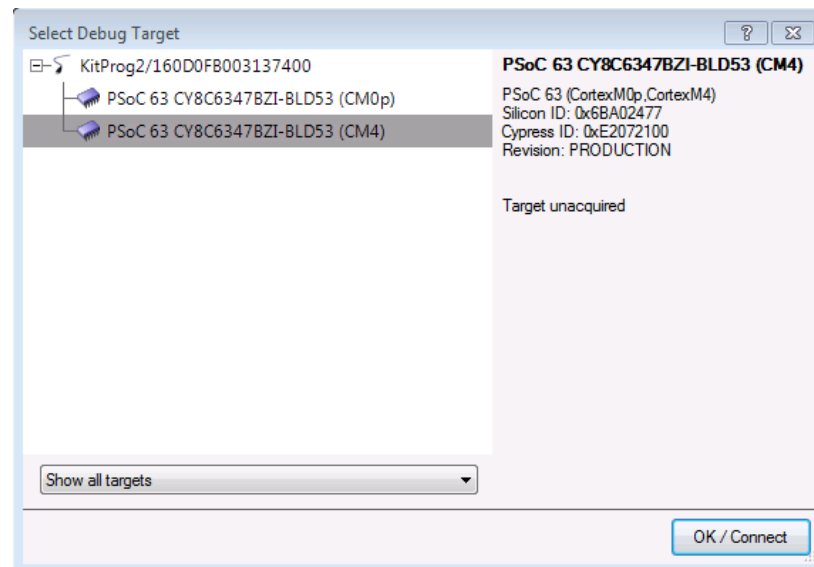


## Operation

Do the following to execute the code example project. See the [Design and Implementation](#) section for more details.

1. Connect the CY8CKIT-062-BLE Pioneer Kit to a USB port on your PC. Set the  $V_{DD}$ ; select either 1.8 V or 3.3 V using the switch SW5 on PSoC 6 Pioneer Kit. The SPI/QSPI F-RAM supports wide operating range  $V_{DD} = 1.8$  V to 3.6 V.
2. Open a serial port communication program such as Tera Term or PuTTY and select the corresponding COM port. Configure the terminal to match the UART: 115200 baud rate, 8N1, and Flow control – None. See the [Software Setup](#) section for the Tera Term setup. These settings must match the configuration of the PSoC Creator UART Component in the project.
3. Build and program either project into the CY8CKIT-062-BLE Kit or CY8CKIT-062 Kit, which has serial F-RAM mounted on it. Select the CM4 option for programming, as shown in [Figure 3](#). For more information on building a project or programming a device, see *PSoC Creator Help*.

Figure 3. PSoC 6 MCU Programming (CM4)



4. Observe the UART example header message printed in the terminal window. [Figure 4](#) shows an example of the output.

- If you are using the Excelon™-Ultra (QSPI F-RAM) on the CY8CKIT-062-BLE kit to execute this code example, make sure that the Excelon-Ultra device access mode is set to SPI and the memory and the register latencies are set to 0. See the code example [CE222967 - Excelon™-Ultra QSPI F-RAM Access Using PSoC 6 MCU SMIF](#) to change the access mode and the latency settings in the Excelon-Ultra device using PSoC 6 MCU.

Figure 4. Terminal Output in Tera Term

```

SMIF & DMA initialization finished
*****QSPI F-RAM Access in XIP with PSoC 6 SMIF*****
*****Setting the device access mode to QSPI*****
*****Set the device access mode to QPI*****

8-byte Device ID read in QPI - 0x50 0x51 0x82 0x06 0x7F 0x7F 0x7F 0x7F
8-byte SN read in QPI - 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

::8-Bit Written data::
0x55 0x56 0x57 0x58 0x59 0x5A 0x5B 0x5C 0x5D 0x5E 0x5F 0x60 0x61 0x62 0x63 0x64 0x65 0x66 0x67 0x68 0x69
0x6A 0x6B 0x6C 0x6D 0x6E 0x6F 0x70 0x71 0x72 0x73 0x74 0x75 0x76 0x77 0x78 0x79 0x7A 0x7B 0x7C 0x7D 0x7E
0x7F 0x80 0x81 0x82 0x83 0x84 0x85 0x86 0x87 0x88 0x89 0x8A 0x8B 0x8C 0x8D 0x8E 0x8F 0x90 0x91 0x92 0x93
0x94 0x95 0x96 0x97 0x98 0x99 0x9A 0x9B 0x9C 0x9D 0x9E 0x9F 0xA0 0xA1 0xA2 0xA3 0xA4 0xA5 0xA6 0xA7 0xA8
0xA9 0xAA 0xAB 0xAC 0xAD 0xAE 0xAF 0xB0 0xB1 0xB2 0xB3 0xB4 0xB5 0xB6 0xB7 0xB8 0xB9 0xBA 0xBB 0xBC 0xBD
0xBE 0xBF 0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7 0xC8 0xC9 0xCA 0xCB 0xCC 0xCD 0xCE 0xCF 0xD0 0xD1 0xD2
0xD3 0xD4 0xD5 0xD6 0xD7 0xD8 0xD9 0xDA 0xDB 0xDC 0xDD 0xDE 0xDF 0xE0 0xE1 0xE2 0xE3 0xE4 0xE5 0xE6 0xE7
0xE8 0xE9 0xEA 0xEB 0xEC 0xED 0xEE 0xEF 0xF0 0xF1 0xF2 0xF3 0xF4 0xF5 0xF6 0xF7 0xF8 0xF9 0xFA 0xFB 0xFC
0xFD 0xFE 0xFF 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F 0x10 0x11
0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1A 0x1B 0x1C 0x1D 0x1E 0x1F 0x20 0x21 0x22 0x23 0x24 0x25 0x26
0x27 0x28 0x29 0x2A 0x2B 0x2C 0x2D 0x2E 0x2F 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x3A 0x3B
0x3C 0x3D 0x3E 0x3F 0x40 0x41 0x42 0x43 0x44 0x45 0x46 0x47 0x48 0x49 0x4A 0x4B 0x4C 0x4D 0x4E 0x4F 0x50
0x51 0x52 0x53

::8-Bit Read data in QPI normal mode::
0x55 0x56 0x57 0x58 0x59 0x5A 0x5B 0x5C 0x5D 0x5E 0x5F 0x60 0x61 0x62 0x63 0x64 0x65 0x66 0x67 0x68 0x69
0x6A 0x6B 0x6C 0x6D 0x6E 0x6F 0x70 0x71 0x72 0x73 0x74 0x75 0x76 0x77 0x78 0x79 0x7A 0x7B 0x7C 0x7D 0x7E
0x7F 0x80 0x81 0x82 0x83 0x84 0x85 0x86 0x87 0x88 0x89 0x8A 0x8B 0x8C 0x8D 0x8E 0x8F 0x90 0x91 0x92 0x93
0x94 0x95 0x96 0x97 0x98 0x99 0x9A 0x9B 0x9C 0x9D 0x9E 0x9F 0xA0 0xA1 0xA2 0xA3 0xA4 0xA5 0xA6 0xA7 0xA8
0xA9 0xAA 0xAB 0xAC 0xAD 0xAE 0xAF 0xB0 0xB1 0xB2 0xB3 0xB4 0xB5 0xB6 0xB7 0xB8 0xB9 0xBA 0xBB 0xBC 0xBD
0xBE 0xBF 0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7 0xC8 0xC9 0xCA 0xCB 0xCC 0xCD 0xCE 0xCF 0xD0 0xD1 0xD2
0xD3 0xD4 0xD5 0xD6 0xD7 0xD8 0xD9 0xDA 0xDB 0xDC 0xDD 0xDE 0xDF 0xE0 0xE1 0xE2 0xE3 0xE4 0xE5 0xE6 0xE7
0xE8 0xE9 0xEA 0xEB 0xEC 0xED 0xEE 0xEF 0xF0 0xF1 0xF2 0xF3 0xF4 0xF5 0xF6 0xF7 0xF8 0xF9 0xFA 0xFB 0xFC
0xFD 0xFE 0xFF 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F 0x10 0x11
0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1A 0x1B 0x1C 0x1D 0x1E 0x1F 0x20 0x21 0x22 0x23 0x24 0x25 0x26
0x27 0x28 0x29 0x2A 0x2B 0x2C 0x2D 0x2E 0x2F 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x3A 0x3B
0x3C 0x3D 0x3E 0x3F 0x40 0x41 0x42 0x43 0x44 0x45 0x46 0x47 0x48 0x49 0x4A 0x4B 0x4C 0x4D 0x4E 0x4F 0x50
0x51 0x52 0x53

::DMA operations beginning::
DMA: Transferring Data
DMA: Done

::SRAM BUFFER after DMA transfer::
0x11111111 0x22222222 0x33333333 0x44444444 0x55555555 0x66666666 0x77777777 0x88888888 0x99999999
0xaaaaaaaa 0xbbbbbbbb 0xcccccccc 0xdddddddd 0xeeeeeeee 0xffffffff 0x11111110 0x22222221 0x33333332
0x44444443 0x55555554 0x66666665 0x77777776 0x88888887 0x99999998 0xaaaaaaaa

::XIP Buffer data in SRAM after DMA transfer::
0x11111111 0x22222222 0x33333333 0x44444444 0x55555555 0x66666666 0x77777777 0x88888888 0x99999999
0xaaaaaaaa 0xbbbbbbbb 0xcccccccc 0xdddddddd 0xeeeeeeee 0xffffffff 0x11111110 0x22222221 0x33333332
0x44444443 0x55555554 0x66666665 0x77777776 0x88888887 0x99999998 0xaaaaaaaa

::XIP buffer after multiply in external memory::
0x22222222 0x44444444 0x66666666 0x88888888 0xaaaaaaaa 0xcccccccc 0xeeeeeeee 0x11111110 0x33333332
0x55555554 0x77777776 0x99999998 0xbbbbbbbb 0xdddddddd 0xffffffff 0x22222220 0x44444442 0x66666664
0x88888886 0xaaaaaaaa 0xcccccccc 0xeeeeeeee 0x11111110 0x33333330 0x55555552
::DONE::
  
```

## Design and Implementation

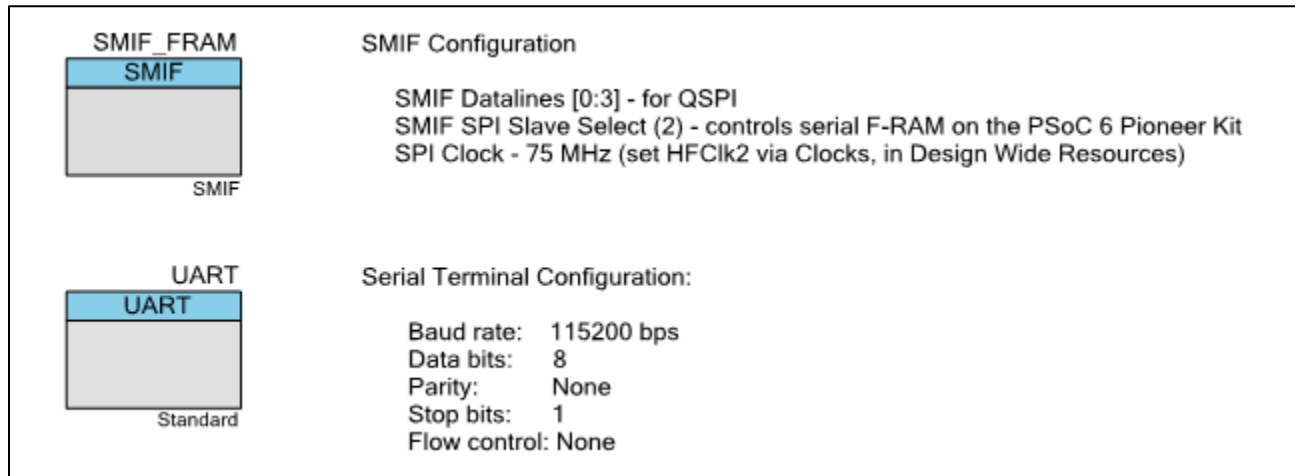
### FRAM\_XIP Project

Figure 5 shows the top level schematic with the Components for this project. An instance of the SMIF Component enables communication with the QSPI-based F-RAM. The SMIF Component is configured in Quad-SPI mode with 4 data lines, a single slave select line, and the SPI clock at 75 MHz. The UART Component prints out the data being written to and read from the F-RAM. The UART also prints information about any failures that occur during operation.

The SMIF Component provides all of the functions required to communicate with the F-RAM in the MMIO mode (non-XIP). On a device reset, the firmware starts up the SMIF in MMIO mode and reads the device ID and serial number from the F-RAM to validate that it is the correct device. On successful device ID checks, the firmware performs an MMIO mode write and read to validate that the memory is working as intended before transitioning to XIP mode.

After the MMIO write and reads are complete, the SMIF is transitioned to XIP mode. In this mode, the firmware transfers data between 8-bit and 32-bit arrays in SRAM and the external F-RAM.

Figure 5. F-RAM XIP Schematic



## FRAM\_XIP Components and Settings

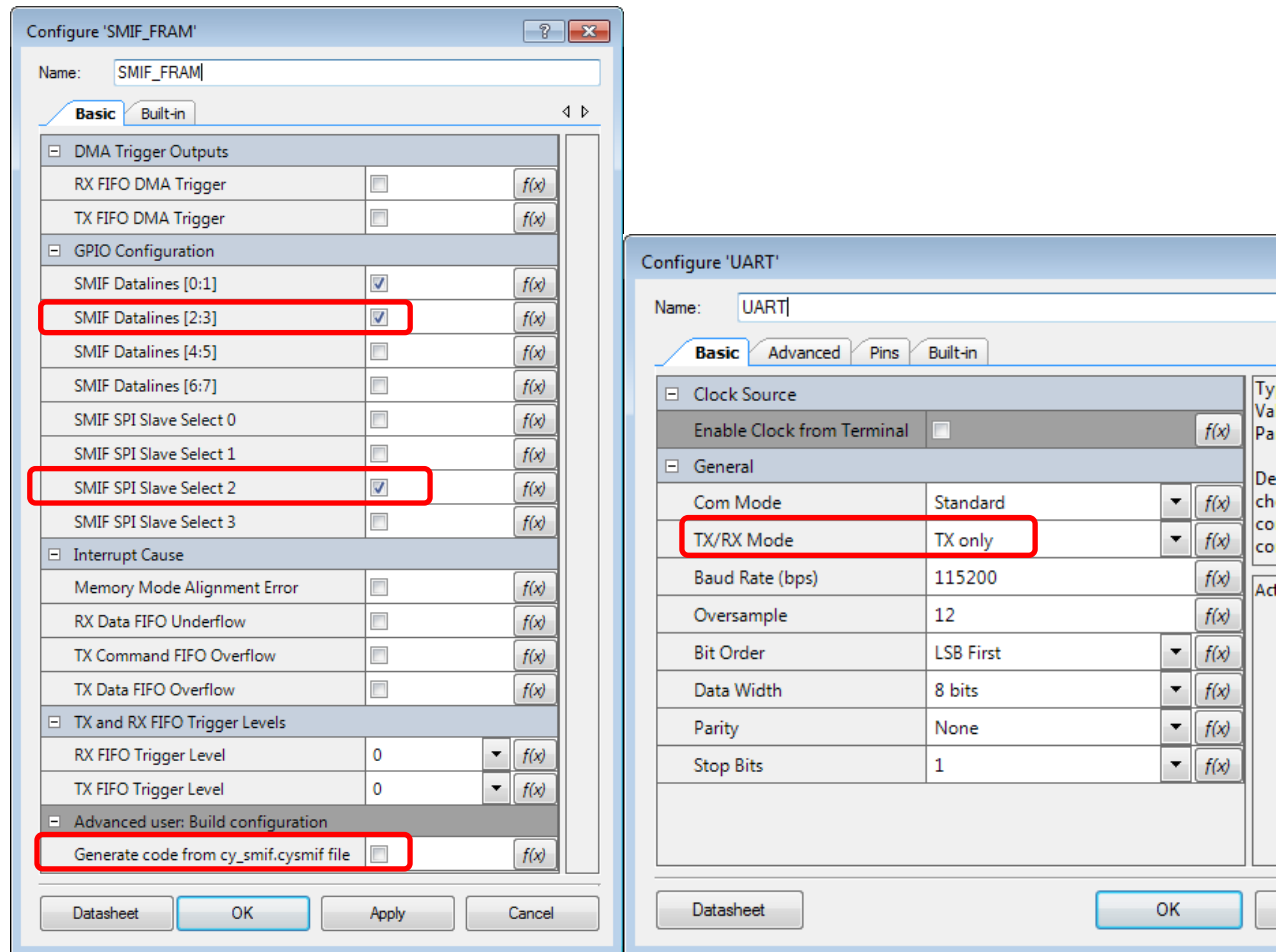
Table 1 lists the PSoC Creator Components used in this example and how they are used in the design.

Table 1. PSoC Creator Components for the FRAM\_XIP Project

Component	Instance Name	Purpose
SMIF	SMIF_FRAM	Enables SPI-based communication with an external memory
UART (SCB)	UART	Enables communication with a serial terminal

The Component configuration settings are shown in the following figures. Figure 6 shows the configuration settings for the SMIF and UART Components.

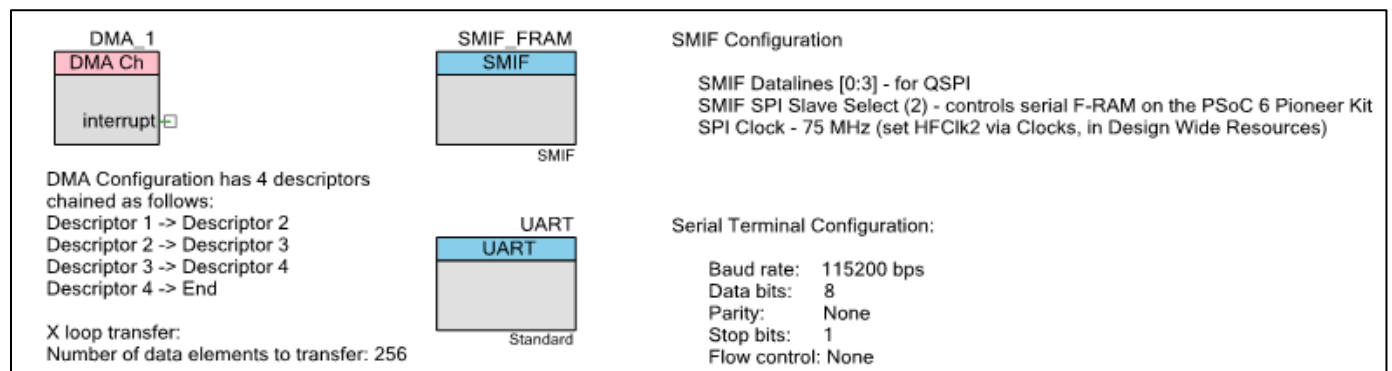
Figure 5. SMIF and UART Component Configuration



## FRAM\_XIP DMA Project

This project follows the same functionality as the FRAM\_XIP project, however, it uses DMA to transfer the data from SRAM to F-RAM. The DMA initializes four descriptors at the addresses of each of the data arrays. On a firmware trigger, the DMA transfers data from the local 32-bit array to an external array that the SMIF Component has mapped into the CPU memory space. To demonstrate manipulation of the data in XIP mode, this data is then doubled and printed over the UART. Figure 7 shows the top level schematic and Components used to implement this project.

Figure 7. FRAM\_XIP\_DMA Schematic



## Components and Settings

Table 2 lists the PSoC Creator Components used in this example and how they are used in the design.

Table 2. PSoC Creator Components

Component	Instance Name	Purpose
SMIF	SMIF_FRAM	Enables SPI-based communication with an external memory
UART (SCB)	UART	Enables communication with a serial terminal
DMA	DMA_1	Enables direct memory access

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

The Component configuration settings are shown in the following figures. Figure 6 shows the configuration settings for the SMIF and UART Components.

Figure 6. SMIF and UART Component Configuration

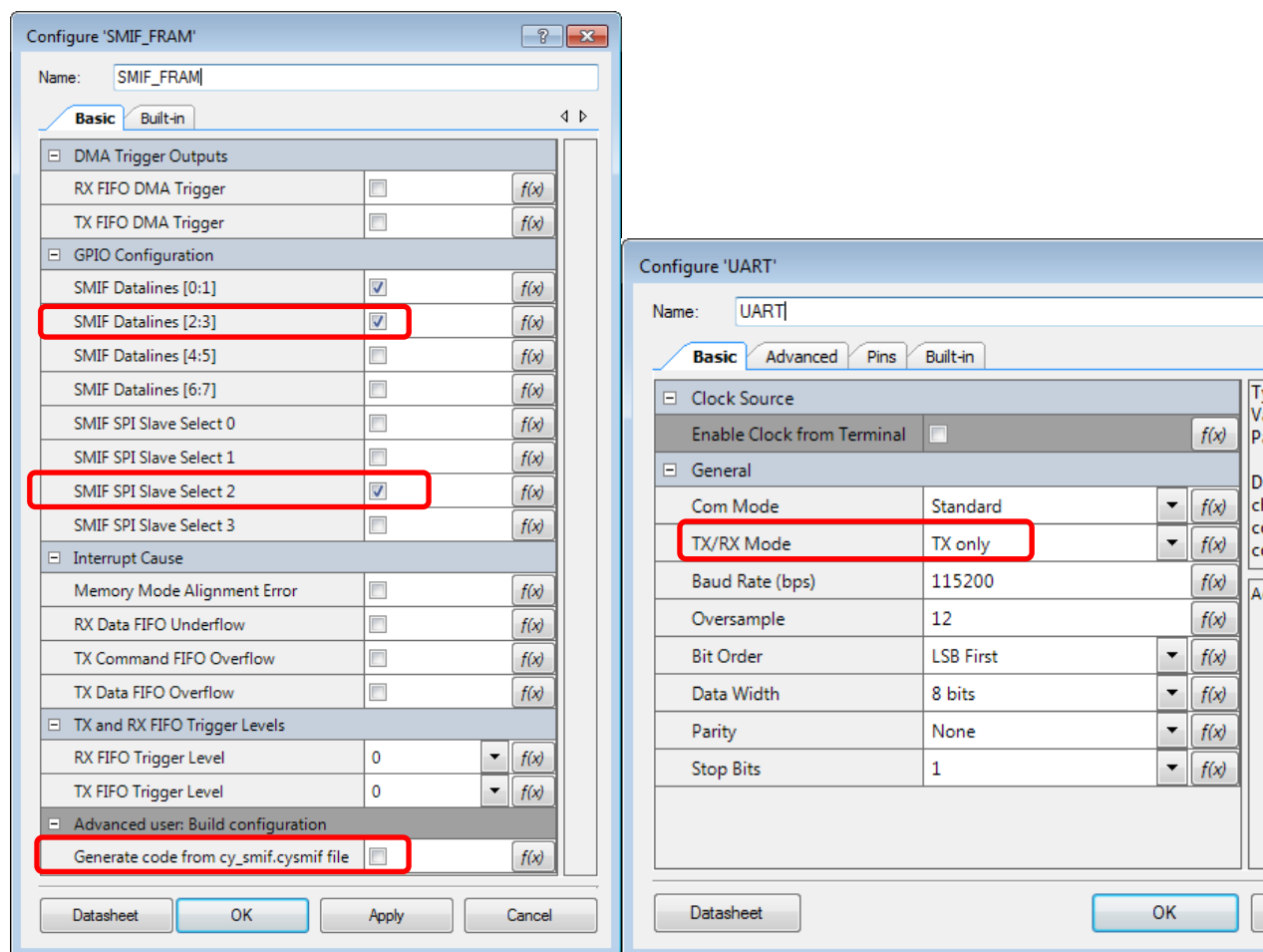
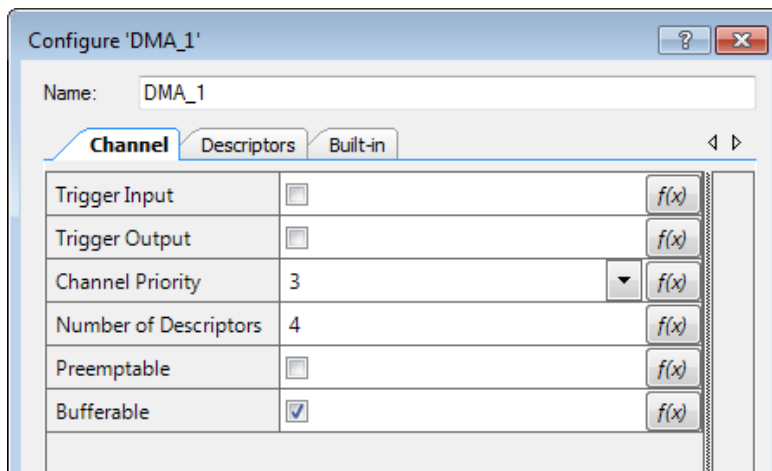


Figure 7. DMA Settings



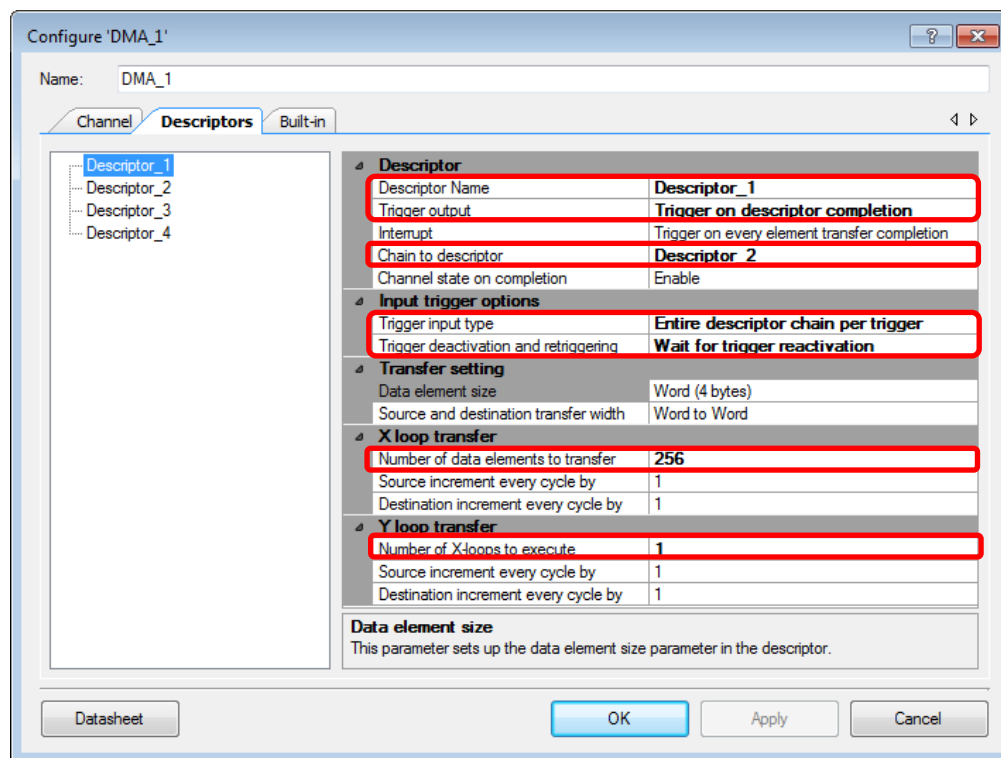
Configure 'DMA\_1'

Name: DMA\_1

Channel Descriptors Built-in

Trigger Input	<input type="checkbox"/>	f(x)
Trigger Output	<input type="checkbox"/>	f(x)
Channel Priority	3	f(x)
Number of Descriptors	4	f(x)
Preemptable	<input type="checkbox"/>	f(x)
Bufferable	<input checked="" type="checkbox"/>	f(x)

Figure 8. DMA Descriptor Settings



Configure 'DMA\_1'

Name: DMA\_1

Channel Descriptors Built-in

Descriptor\_1  
Descriptor\_2  
Descriptor\_3  
Descriptor\_4

**Descriptor**

Descriptor Name	Descriptor_1
Trigger output	Trigger on descriptor completion
Interrupt	Trigger on every element transfer completion
Chain to descriptor	Descriptor_2
Channel state on completion	Enable

**Input trigger options**

Trigger input type	Entire descriptor chain per trigger
Trigger deactivation and retriggering	Wait for trigger reactivation

**Transfer setting**

Data element size	Word (4 bytes)
Source and destination transfer width	Word to Word

**X loop transfer**

Number of data elements to transfer	256
Source increment every cycle by	1
Destination increment every cycle by	1

**Y loop transfer**

Number of X-loops to execute	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.

Datasheet OK Apply Cancel

Configure 'DMA\_1'
 

Name: DMA\_1

Channel Descriptors Built-in

Descriptor\_1  
 Descriptor\_2  
 Descriptor\_3  
 Descriptor\_4

**Descriptor**

Descriptor Name: **Descriptor\_2**

Trigger output: **Trigger on descriptor completion**

Interrupt: Trigger on every element transfer completion

Chain to descriptor: **Descriptor\_3**

Channel state on completion: Enable

**Input trigger options**

Trigger input type: **Entire descriptor chain per trigger**

Trigger deactivation and retriggering: **Wait for trigger reactivation**

**Transfer setting**

Data element size: Word (4 bytes)

Source and destination transfer width: Word to Word

**X loop transfer**

Number of data elements to transfer: **256**

Source increment every cycle by: 1

Destination increment every cycle by: 1

**Y loop transfer**

Number of X-loops to execute: **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.

Datasheet OK Apply Cancel

Configure 'DMA\_1'
 

Name: DMA\_1

Channel Descriptors Built-in

Descriptor\_1  
 Descriptor\_2  
 Descriptor\_3  
 Descriptor\_4

**Descriptor**

Descriptor Name: **Descriptor\_3**

Trigger output: **Trigger on descriptor completion**

Interrupt: Trigger on every element transfer completion

Chain to descriptor: **Descriptor\_4**

Channel state on completion: Enable

**Input trigger options**

Trigger input type: **Entire descriptor chain per trigger**

Trigger deactivation and retriggering: **Wait for trigger reactivation**

**Transfer setting**

Data element size: Word (4 bytes)

Source and destination transfer width: Word to Word

**X loop transfer**

Number of data elements to transfer: **256**

Source increment every cycle by: 1

Destination increment every cycle by: 1

**Y loop transfer**

Number of X-loops to execute: **1**

Source increment every cycle by: 1

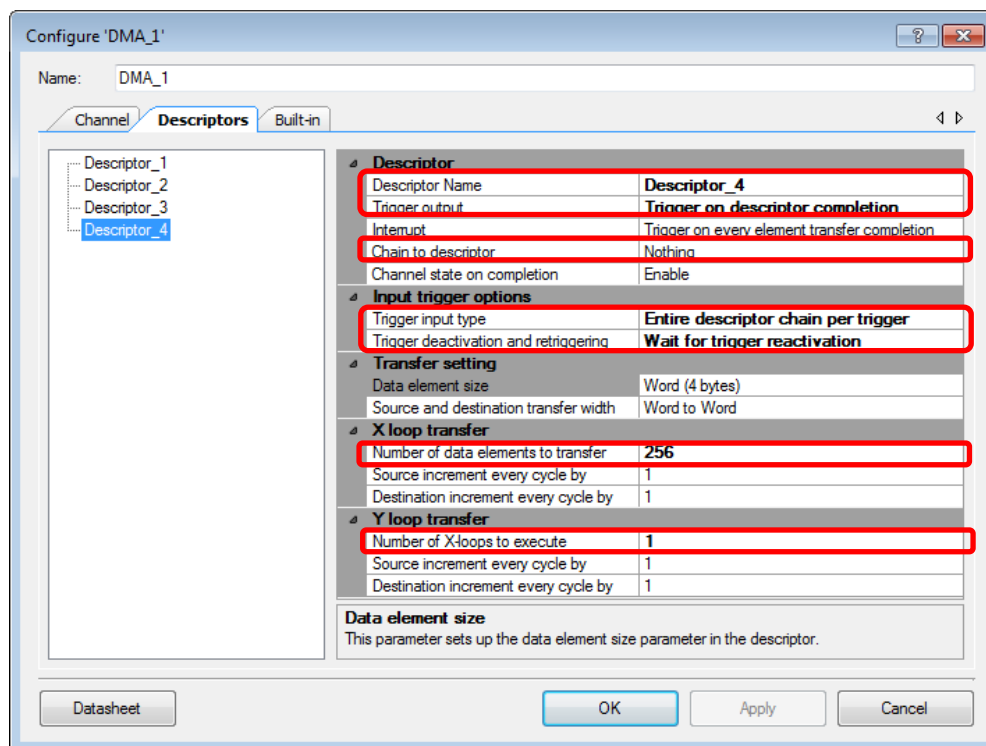
Destination increment every cycle by: 1

**Data element size**

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

Datasheet OK Apply Cancel

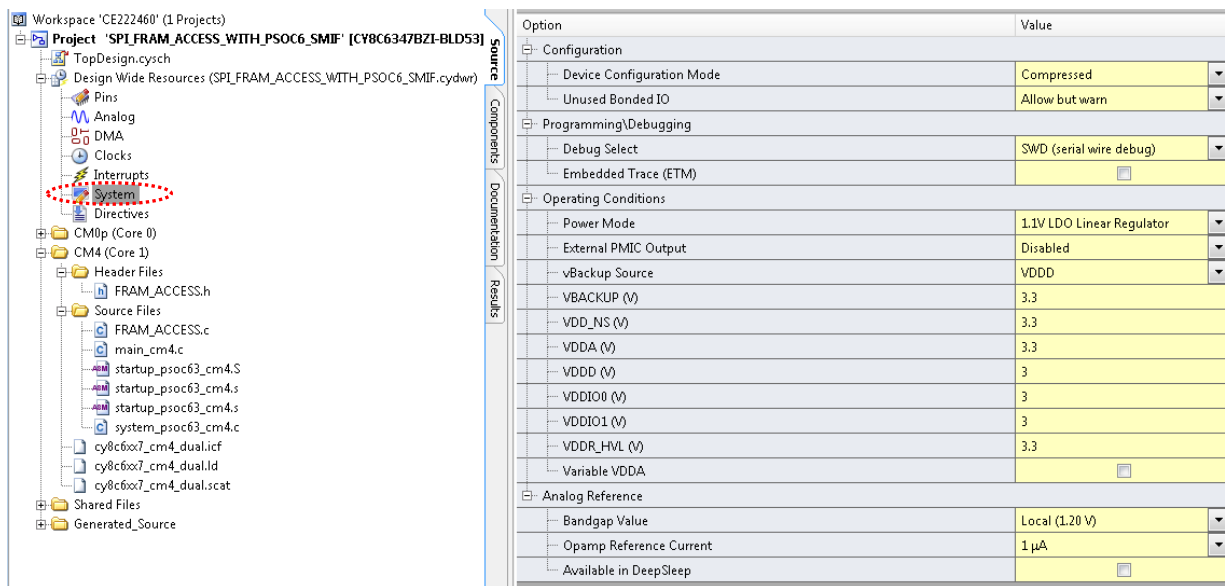




## Design-Wide Resources

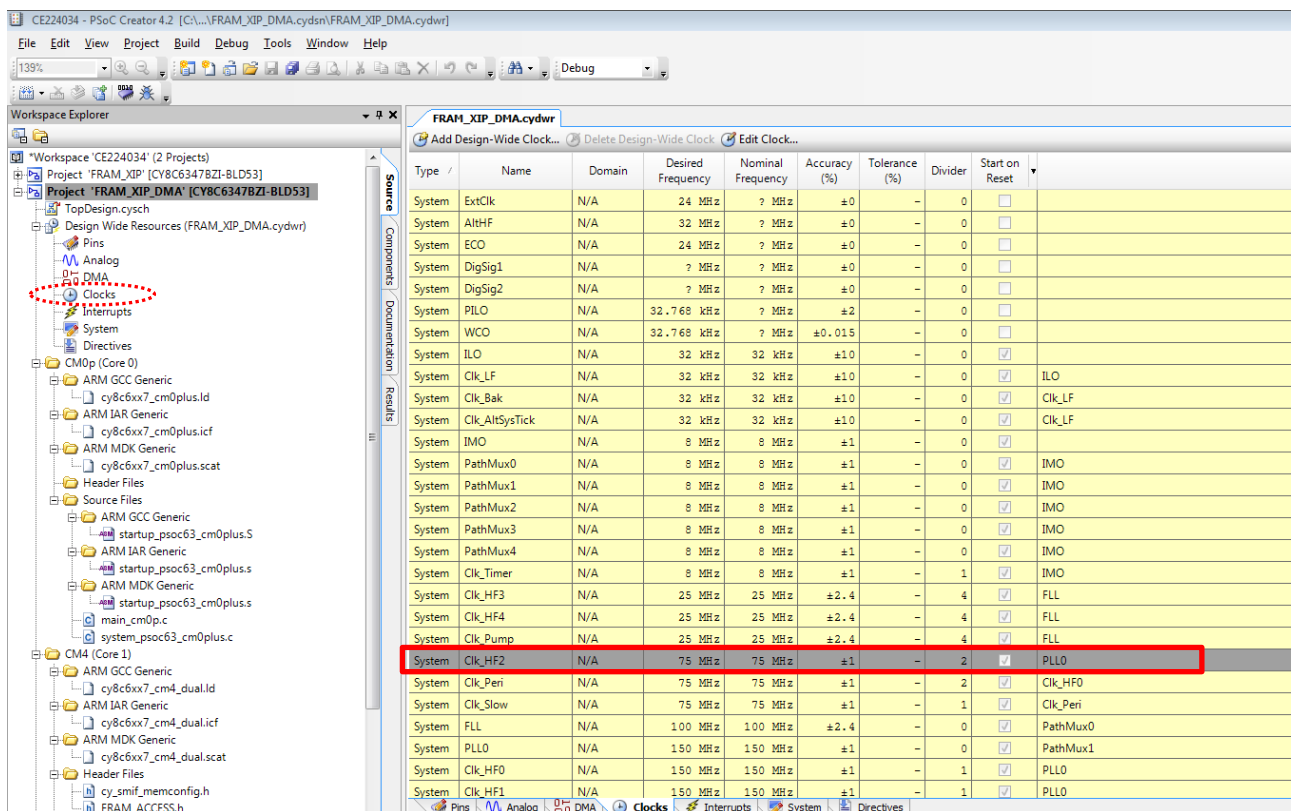
Make sure that  $V_{DD}$  (**PSoC Creator > Design Wide Resources > System** tab) is set correctly to match the SPI F-RAM operating range ( $V_{DD}/V_{CC}$ ), as shown in Figure 9.

Figure 9.  $V_{DD}$  Setting using Design Wide Resources



Make sure that SMIF SPI clock frequency is set at 80 MHz or below. To change the SMIF clock frequency, go to **PSoC Creator > Design Wide Resources**, and double-click **Clocks**, as shown in Figure 10.

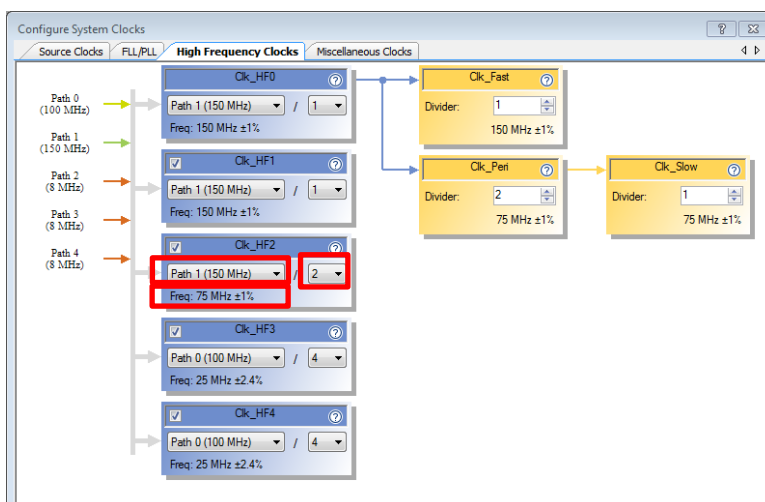
Figure 10. SMIF Clock Setting Using Design Wide Resources – Step 1



Double-click anywhere in the **Cik\_HF2** row, as highlighted in Figure 10. A new **Configure System Clocks** window opens as shown in Figure 11.

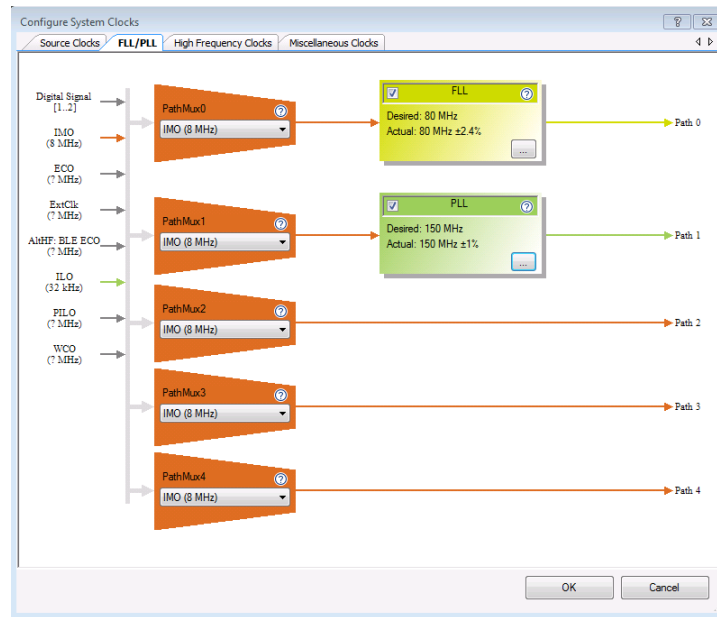
Go to the **High Frequency Clocks** tab and select the appropriate clock path and the clock divider from corresponding drop-down lists to set the frequency to 80 MHz or below (see Figure 11). This code example sets the SPI clock frequency to 75 MHz. Select **OK**.

Figure 11. SMIF SPI Clock Setting using Design Wide Resources – Step 2



You can use FLL/PLL, as shown in Figure 12, to configure other frequency options for Path 0.

Figure 12. FLL/PLL Setting examples – Step 3



### PSoC 6 Pin Assignment

Figure 13 shows the pin assignment for the code example. The following PSoC 6 MCU pins control the respective SPI F-RAM control pins for the SPI communication.

Figure 13. PSoC 6 MCU Pin Assignments for Code Example

	Name	Port	Pin	Lock
<input checked="" type="checkbox"/>	\SMIF_FRAM:spi_clk\	P11[7]	A5	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	\SMIF_FRAM:spi_data_0\	P11[6]	B5	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	\SMIF_FRAM:spi_data_1\	P11[5]	A6	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	\SMIF_FRAM:spi_data_2\	P11[4]	B6	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	\SMIF_FRAM:spi_data_3\	P11[3]	C6	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	\SMIF_FRAM:spi_select2\	P11[0]	F5	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	\UART:rx\	P5[0]	L6	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	\UART:tx\	P5[1]	K6	<input checked="" type="checkbox"/>

### Reusing This Example

This example is designed for the CY8CKIT-062-BLE Pioneer Kit with a serial F-RAM device mounted. To port the design to a different PSoC 6 MCU device, kit, or both, change the target device using the Device Selector and update the pin assignments in Design Wide Resources Pins settings as needed. For single-core PSoC 6 MCU devices, port the code from *main\_cm4.c* to *main.c*.

This code example also includes QSPI F-RAM-specific PSoC 6 MCU SMIF driver files (.c and .h) that can be used in another project. The driver file description is as follows. For more details on driver files content, see the code example project.

- **CY\_SMIF\_FRAM\_CONFIG.h** - This file contains all defines for PSoC 6 MCU SMIF Component to access the QSPI F-RAM in XIP mode.
- **CY\_SMIF\_FRAM\_CONFIG.c** - This file contains the initialization for PSoC 6 MCU SMIF Component to access the QSPI F-RAM in XIP mode
- **FRAM\_ACCESS.h** - This file contains variable and all API declarations to access the QSPI F-RAM in PSoC 6 MCU SMIF MMIO mode.
- **FRAM\_ACCESS.c** - This file contains all API definitions to access the QSPI F-RAM in PSoC 6 MCU SMIF MMIO mode.

In some cases, a resource used by a code example (for example, DMA) is not supported on another device; in such cases, the example will not work. If you build the code targeted at such a device, you will get errors. See the PSoC device datasheet for information on the supported device.

## Related Documents

Code Examples/ Application Notes	
<a href="#">CE220823 – PSoC 6 MCU SMIF Memory Write and Read Operation</a>	This example demonstrates the write and read operations to the Serial Memory Interface (SMIF) in PSoC 6 MCU.
<a href="#">CE222460 - SPI F-RAM Access Using PSoC 6 MCU SMIF</a>	CE222460 provides a code example that implements the SPI host controller on PSoC 6 MCU using the SMIF Component and demonstrates accessing different features of the SPI F-RAM.
<a href="#">CE224073 – SPI F-RAM Access Using PSoC 6 MCU SMIF in Memory Mapped (XIP) Mode</a>	SPI F-RAM Access Using PSoC 6 MCU SMIF in Memory Mapped (XIP) Mode
<a href="#">AN304 – SPI Guide for F-RAM™</a>	AN304 provides the functional description, timing, and example code for SPI F-RAMs.
PSoC Creator Component Datasheets	
<a href="#">UART</a>	UART communications interface
<a href="#">SMIF</a>	Serial Memory Interface
<a href="#">Direct Memory Access</a>	Supports data transfer independent of the CPU
Device Documentation	
<a href="#">PSoC 6 MCU: PSoC 63 with BLE Datasheet</a>	<a href="#">PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual</a>
<a href="#">EXCELON-ULTRA 4-MBIT (512K X 8) QUAD SPI F-RAM Datasheet</a>	
Development Kit Documentation	
<a href="#">CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit</a>	

## Document History

Document Title: CE224034 – QSPI F-RAM Access Using PSoC 6 MCU with SMIF in XIP Mode

Document Number: 002-24034

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6267846	BFMC	04/30/2019	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	<a href="http://cypress.com/arm">cypress.com/arm</a>
Automotive	<a href="http://cypress.com/automotive">cypress.com/automotive</a>
Clocks & Buffers	<a href="http://cypress.com/clocks">cypress.com/clocks</a>
Interface	<a href="http://cypress.com/interface">cypress.com/interface</a>
Internet of Things	<a href="http://cypress.com/iot">cypress.com/iot</a>
Memory	<a href="http://cypress.com/memory">cypress.com/memory</a>
Microcontrollers	<a href="http://cypress.com/mcu">cypress.com/mcu</a>
PSoC	<a href="http://cypress.com/psoc">cypress.com/psoc</a>
Power Management ICs	<a href="http://cypress.com/pmic">cypress.com/pmic</a>
Touch Sensing	<a href="http://cypress.com/touch">cypress.com/touch</a>
USB Controllers	<a href="http://cypress.com/usb">cypress.com/usb</a>
Wireless Connectivity	<a href="http://cypress.com/wireless">cypress.com/wireless</a>

### PSoC® Solutions

[PSoC 1](#) | [PSoC 3](#) | [PSoC 4](#) | [PSoC 5LP](#) | [PSoC 6 MCU](#)

### Cypress Developer Community

[Community Forums](#) | [Projects](#) | [Videos](#) | [Blogs](#) | [Training](#) | [Components](#)

### Technical Support

[cypress.com/support](http://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, 2019. This document is the property of Cypress Semiconductor Corporation and its subsidiaries ("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 shall have no liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. CYPRESS DOES NOT REPRESENT, WARRANT, OR GUARANTEE THAT CYPRESS PRODUCTS, OR SYSTEMS CREATED USING CYPRESS PRODUCTS, WILL BE FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (collectively, "Security Breach"). Cypress disclaims any liability relating to any Security Breach, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from any Security Breach. 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. "High-Risk Device" means any device or system whose failure could cause personal injury, death, or property damage. Examples of High-Risk Devices are weapons, nuclear installations, surgical implants, and other medical devices. "Critical Component" means any component of a High-Risk Device whose failure to perform can be reasonably expected to cause, directly or indirectly, the failure of the High-Risk Device, 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 any use of a Cypress product as a Critical Component in a High-Risk Device. You shall indemnify and hold Cypress, its directors, officers, employees, agents, affiliates, distributors, and assigns harmless from and against all claims, costs, damages, and expenses, arising out of any claim, including claims for product liability, personal injury or death, or property damage arising from any use of a Cypress product as a Critical Component in a High-Risk Device. Cypress products are not intended or authorized for use as a Critical Component in any High-Risk Device except to the limited extent that (i) Cypress's published data sheet for the product explicitly states Cypress has qualified the product for use in a specific High-Risk Device, or (ii) Cypress has given you advance written authorization to use the product as a Critical Component in the specific High-Risk Device and you have signed a separate indemnification agreement.

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](http://cypress.com). Other names and brands may be claimed as property of their respective owners.