# CE222967 – Excelon-Ultra QSPI F-RAM Access Using PSoC 6 MCU SMIF

## Objective

CE222967 demonstrates accessing the Excelon™-Ultra QSPI F-RAM™ using PSoC® 6 MCU's Serial Memory Interface (SMIF) Component in memory mapped I/O (MMIO) mode.

## Overview

CE222967 provides a code example that implements the QSPI host controller on the PSoC 6 MCU device using the SMIF Component and demonstrates accessing different features of the QSPI F-RAM. The result is displayed by driving the status LED (RGB), which turns green when the result is a pass, and turns red when the result is a fail. The code example also enables the UART interface to connect to a PC to monitor the result.

## Requirements

**Tool:** PSoC Creator™ 4.2; Peripheral Driver Library (PDL) 3.1.0

**Programming Language:** C (Arm® GCC 5.4-2016-q2-update, Arm MDK 5.22)

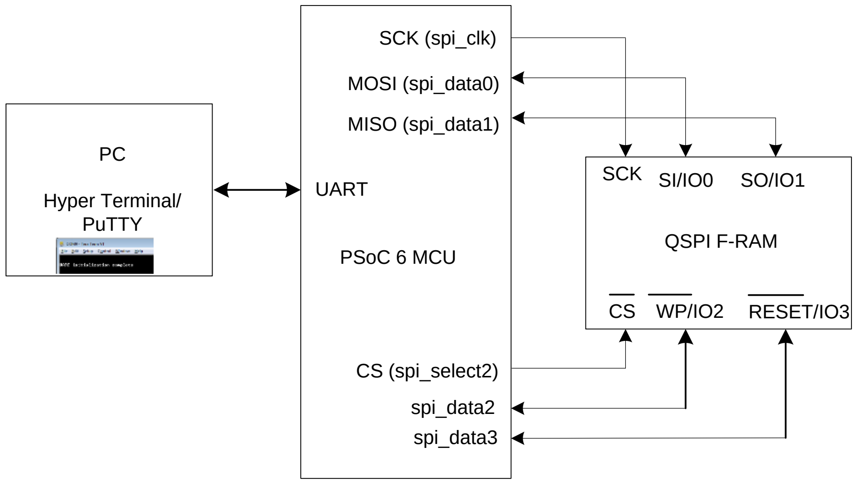
**Associated Parts:** All PSoC 6 MCU parts

**Related Hardware:** PSoC 6 WiFi-BT Pioneer Kit (CY8CKIT-062-WiFi-BT)

## Hardware Setup

The hardware setup includes connecting the QSPI F-RAM with PSoC 6 MCU. You can use either dedicated hardware as described in the Requirements section or can connect via jumper wires by tapping the SMIF QSPI control pins and connect to the QSPI pins of an external QSPI F-RAM. This example uses the PSoC 6 WiFi-BT Pioneer kit's default configuration. See the kit guide to ensure the kit is configured correctly.

Figure 1. Hardware Setup Block Diagram

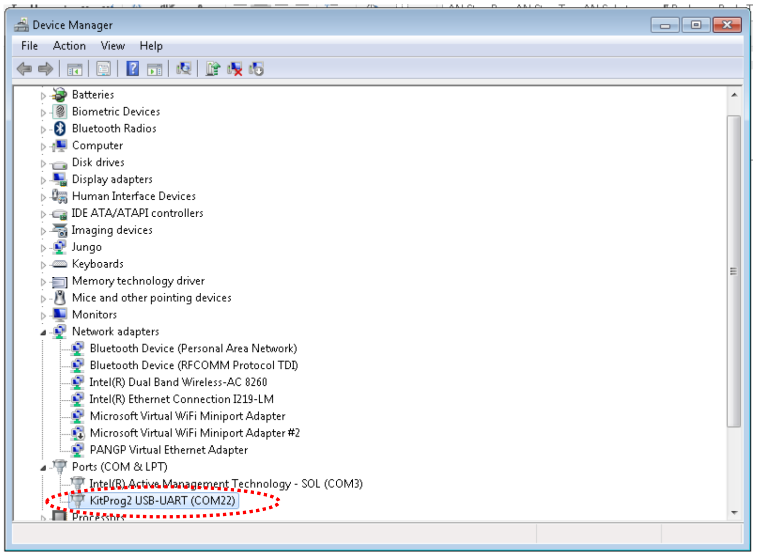


## Software Setup

This section demonstrates the procedure to set up a serial (UART) connection using PuTTY on a PC to communicate with the PSoC 6 Pioneer Kit. PuTTY is a free SSH and Telnet client for Windows. You can download PuTTY from www.putty.org. Follow these instructions to determine the COM port number and set up PuTTY to monitor the code example outputs on your PC.

1. Connect the PSoC 6 Pioneer Kit to the PC using the USB cable. The kit enumerates as KitProg2 USB-UART and is available under the **Device Manager** > **Ports (COM & LPT)**. A communication port (COMx) is assigned to KitProg2 USB-UART; for example, COM22 is assigned to PSoC 6 Pioneer Kit on the sample setup, shown in Figure 2.

Figure 2. KitProg2 USB-UART in Device Manager


Device Manager
File
Action
View
Help
Batteries
Biometric Devices
Bluetooth Radios
Computer
Disk drives
Display adapters
Human Interface Devices
IDE ATA/ATAPI controllers
Imaging devices
Jungo
Keyboards
Memory technology driver
Mice and other pointing devices
Monitors
Network adapters
Bluetooth Device (Personal Area Network)
Bluetooth Device (RFCOMM Protocol TDI)
Intel(R) Dual Band Wireless-AC 8260
Intel(R) Ethernet Connection I219-LM
Microsoft Virtual WiFi Miniport Adapter
Microsoft Virtual WiFi Miniport Adapter #2
PANGP Virtual Ethernet Adapter
Ports (COM & LPT)
Intel(R) Active Management Technology - SOL (COM3)
KitProg2 USB-UART (COM22)
Processors

2. After you download and install PuTTY, double-click the PuTTY icon and select **Serial** under **Connection**.

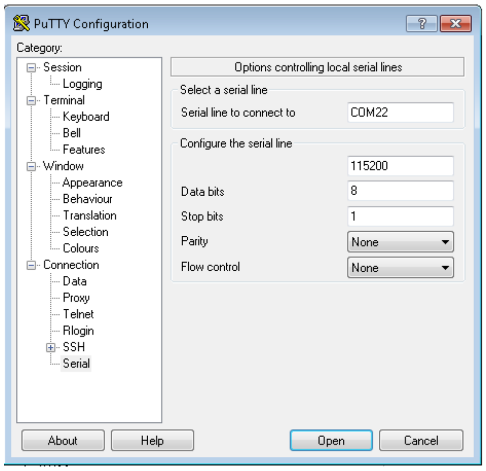
3. A new window opens where you can select the communication port. Do the following in the **Options controlling local serial lines** section:

   a. Enter the PSoC 6 Port (COM & LPT), COMx, in **Serial line to connect to**. This code example uses **COM22**. Verify the COM setting for your setup and select the appropriate COMx.

   b. Enter **Speed (baud)**, **Data bits**, and **Stop bits**.

   c. Select **Parity** and **Flow control**.

Figure 3. Open New Connection



Options controlling local serial lines

Select a serial line

Serial line to connect to: COM22

Configure the serial line

115200

Data bits: 8

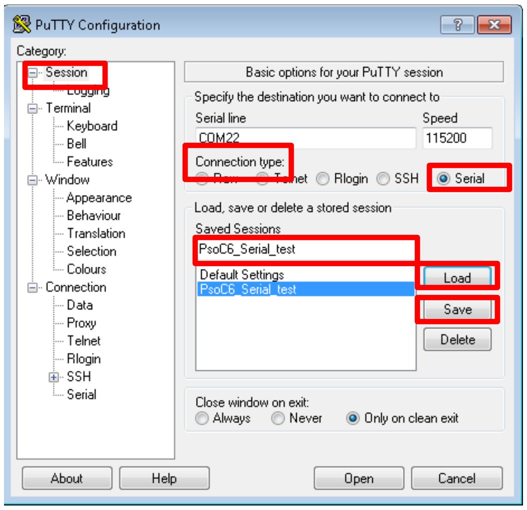
Stop bits: 1

Parity: None

Flow control: None

4. Select **Session** under **Category**. Select **Serial** as the **Connection type** as shown in Figure 4. You can save this current session and load the settings when required. Enter a name in **Saved Sessions** and click **Save**. Click **Open** to proceed.

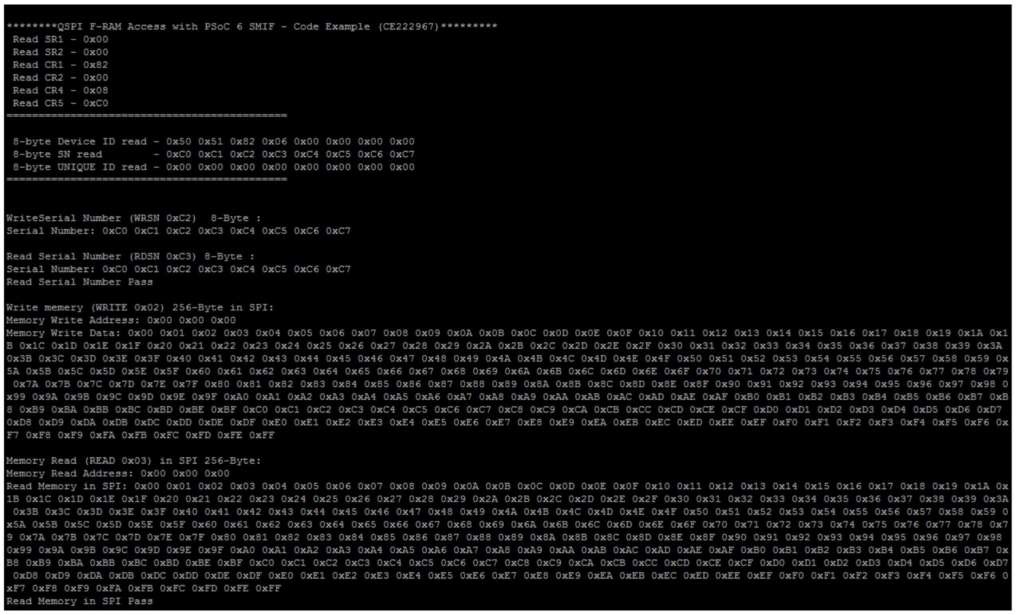
Figure 4. Select Communication Type in PuTTY



Specify the destination you want to connect to
Serial line: COM22
Speed: 115200
Connection type: Raw Telnet Rlogin SSH [✓ Serial]
Load, save or delete a stored session
Saved Sessions: PsoC6_Serial_test
Default Settings
PsoC6_Serial_test
Close window on exit: Always Never [✓ Only on clean exit]

5. The COM terminal window then displays the code example results as shown in Figure 5. You may have to reprogram PSoC 6 MCU with the code example hex file or reset PSoC 6 MCU (already programmed) to restart the code execution and monitor the result.

Figure 5. Result Displayed on PuTTY



```
********QSPI F-RAM Access with PSoC 6 SMIF - Code Example (CE222967)*********
 Read SR1 - 0x00
 Read SR2 - 0x00
 Read CR1 - 0x82
 Read CR2 - 0x00
 Read CR4 - 0x08
 Read CR5 - 0xC0
============================================


 8-byte Device ID read - 0x50 0x51 0x82 0x06 0x00 0x00 0x00 0x00
 8-byte SN read        - 0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7
 8-byte UNIQUE ID read - 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
============================================



WriteSerial Number (WRSN 0xC2)  8-Byte :
Serial Number: 0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7

Read Serial Number (RDSN 0xC3) 8-Byte :
Serial Number: 0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7
Read Serial Number Pass

Write memery (WRITE 0x02) 256-Byte in SPI:
Memory Write Address: 0x00 0x00 0x00
Memory Write Data: 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 0x1
B 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 0x54 0x55 0x56 0x57 0x58 0x59 0x
5A 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 0
x99 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 0xB
8 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 0x
F7 0xF8 0xF9 0xFA 0xFB 0xFC 0xFD 0xFE 0xFF

Memory Read (READ 0x03) in SPI 256-Byte:
Memory Read Address: 0x00 0x00 0x00
Read Memory in SPI: 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 0x
1B 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 0x54 0x55 0x56 0x57 0x58 0x59 0
x5A 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 0x7
9 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 0x
B8 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 0
xF7 0xF8 0xF9 0xFA 0xFB 0xFC 0xFD 0xFE 0xFF
Read Memory in SPI Pass
```

Alternatively, you can run the HyperTerminal if supported on your PC to monitor the above result.

## Operation

This code example demonstrates accessing the QSPI F-RAM's standard write and read features including memory and user registers in Memory Mapped I/O (MMIO) mode. This code example does not support user APIs for execute-in-place (XIP), CRC, and ECC features of F-RAM. The following features of the QSPI F-RAM are demonstrated through this code example:

- Sets the device access mode to default SPI mode. This ensures that the part starts with a known SPI mode.
- Reads two Status and four Configuration registers (SR1, SR2, CR1, CR2, CR4, CR5)
- Reads the 8-byte device ID
- Writes 256 bytes into F-RAM at a given address, in SPI mode
- Reads 256 bytes from F-RAM at a given address, using the READ (0x02) opcode in SPI mode
- Reads 256 bytes from F-RAM at a given address, using the READ (0x02) opcode in DPI mode
- Reads 256 bytes from F-RAM at a given address, using the READ (0x02) opcode in QPI mode
- Reads 256 bytes from F-RAM at a given address, using the FastRead (0x0B) opcode in QPI mode
- Writes and read 256 bytes special sector using SSWR and SSRD commands in QPI mode

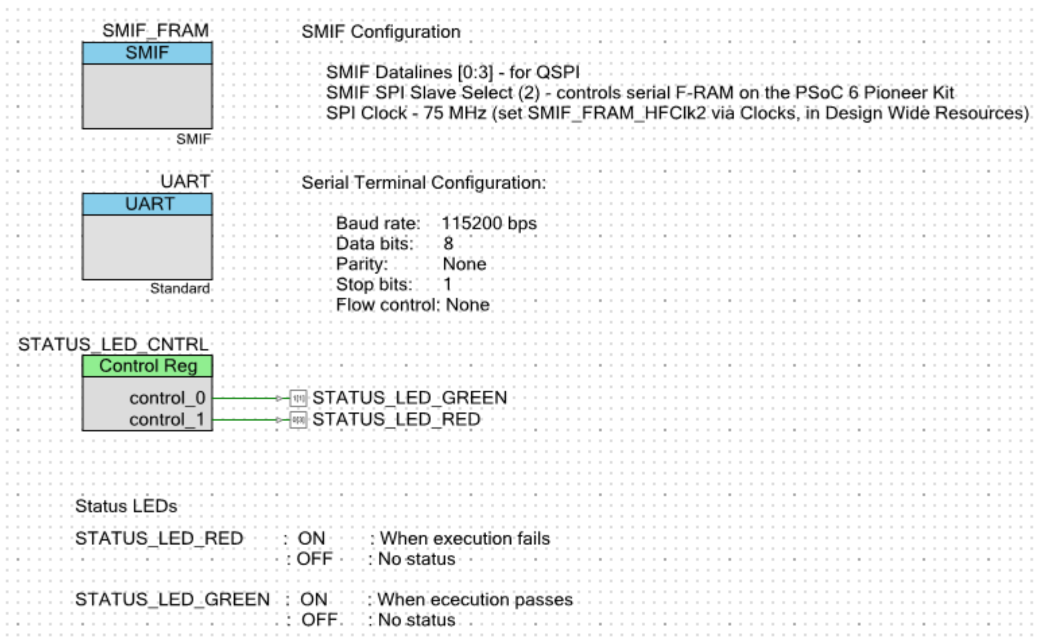
Do the following to execute the code example project. See the Design and Implementation section for more details.

1. Connect the CY8CKIT-062-WiFi-BT Pioneer kit to a USB port on your PC. Set $V_{DD}$ select either 1.8 V or 3.3 V using the switch SW5 on PSoC 6 Pioneer kit. The Excelon-Ultra QSPI F-RAM supports wide operating voltage range $V_{DD}$ = 1.8 V to 3.6 V.
2. Open a serial port communication program such as PuTTY, and select the corresponding COM port. Configure the terminal to match the UART: 115200 baud rate, 8N1, and Flow control – None. These settings must match the configuration of the PSoC Creator UART Component in the project.
3. Build and program the application into the CY8CKIT-062-WiFi-BT Kit or CY8CKIT-062-BLE Kit, which has the QSPI F-RAM mounted on it. For more information on building a project or programming a device, see *PSoC Creator Help*.
4. Observe the result status by monitoring RGB LED. The LED toggles green when result is a pass and red when result is a fail.
5. Observe the UART example header message printed in the terminal window. Figure 5 shows a snapshot of a sample UART terminal output.

## Design and Implementation

Figure 6 shows the design for this code example. The SMIF Component implements quad data lines [Datalines[0:3]) GPIO configuration for interfacing with an external QSPI F-RAM with PSoC 6 MCU. The SMIF Component is configured with four data lines, single slave select line, and the SPI clock (SCK) at 75 MHz. The UART Component outputs debug information to a terminal window. It is configured for 8N1, transmit only, at 115.2 kbps. The code example also uses the Control Register Component to drive the RGB LED on the PSoC 6 Pioneer kit to display results.

Figure 6. CE222967 Design Schematic in PSoC Creator


SMIF_FRAM
SMIF
SMIF
SMIF Configuration
SMIF Datalines [0:3] - for QSPI
SMIF SPI Slave Select (2) - controls serial F-RAM on the PSoC 6 Pioneer Kit
SPI Clock - 75 MHz (set SMIF_FRAM_HFClk2 via Clocks, in Design Wide Resources)
UART
UART
Standard
Serial Terminal Configuration:
Baud rate: 115200 bps
Data bits: 8
Parity: None
Stop bits: 1
Flow control: None
STATUS_LED_CNTRL
Control Reg
control_0
control_1
STATUS_LED_GREEN
STATUS_LED_RED
Status LEDs
STATUS_LED_RED : ON : When execution fails
: OFF : No status
STATUS_LED_GREEN : ON : When ececution passes
: OFF : No status

## Components and Settings

Table 1 lists the PSoC Creator Components used in the three examples.

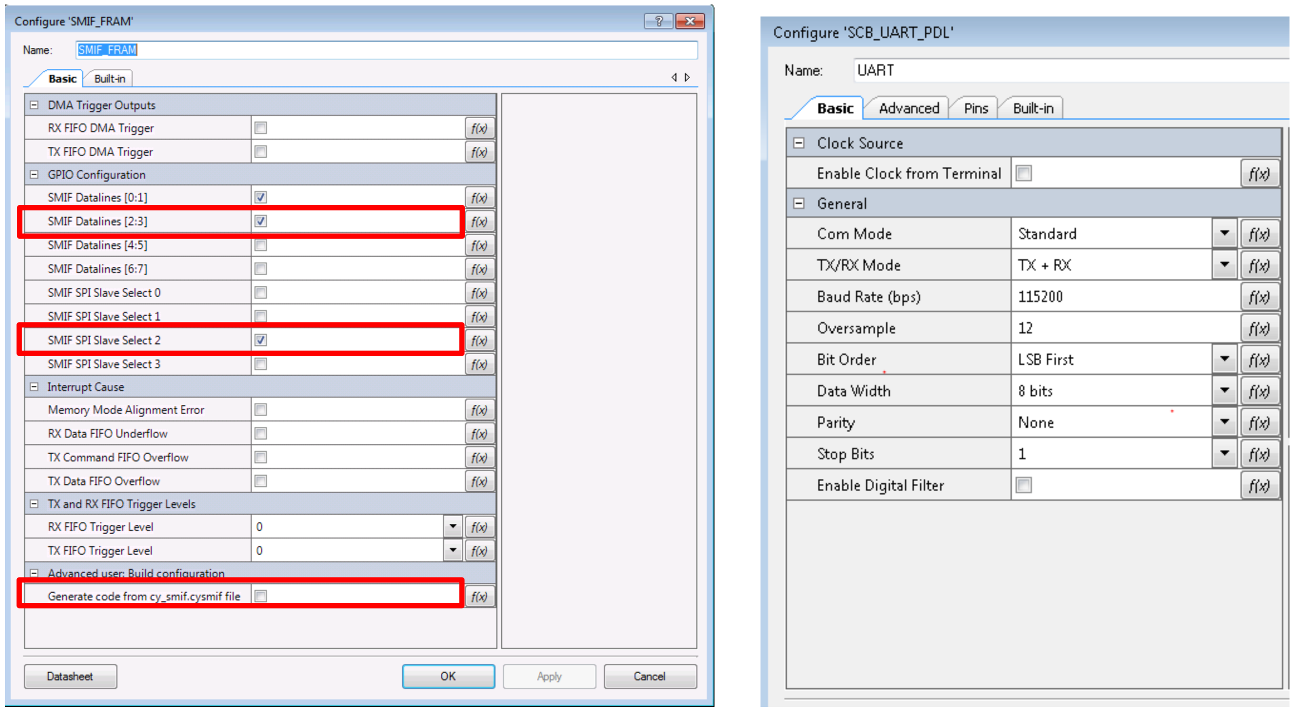
Table 1. PSoC Creator Components

| Component | Instance Name | Purpose |
|---|---|---|
| SMIF(SMIF_PDL) | SMIF_FRAM | The SMIF peripheral block. Configures the QSPI host controller in the design. |
| UART (SCB_UART_PDL) | UART | Handles communication to the terminal window |
| Control Register (CyControlReg) | STATUS_LED_CNTRL | Drives the status (RGB) LED output |

## Parameter Settings

Non-default settings for each Component is outlined in red in the following figures. Figure 7 shows the SMIF_FRAM Component parameter settings.

Figure 7. Settings for SMIF and UART Components (Non-default settings are outlined)



Configure 'SMIF_FRAM'

Name: SMIF_FRAM

Basic | Built-in

| | |
|---|---|
| **DMA Trigger Outputs** | |
| RX FIFO DMA Trigger | ☐ |
| TX FIFO DMA Trigger | ☐ |
| **GPIO Configuration** | |
| SMIF Datalines [0:1] | ✓ |
| SMIF Datalines [2:3] | ✓ |
| SMIF Datalines [4:5] | ☐ |
| SMIF Datalines [6:7] | ☐ |
| SMIF SPI Slave Select 0 | ☐ |
| SMIF SPI Slave Select 1 | ☐ |
| SMIF SPI Slave Select 2 | ✓ |
| SMIF SPI Slave Select 3 | ☐ |
| **Interrupt Cause** | |
| Memory Mode Alignment Error | ☐ |
| RX Data FIFO Underflow | ☐ |
| TX Command FIFO Overflow | ☐ |
| TX Data FIFO Overflow | ☐ |
| **TX and RX FIFO Trigger Levels** | |
| RX FIFO Trigger Level | 0 |
| TX FIFO Trigger Level | 0 |
| **Advanced user: Build configuration** | |
| Generate code from cy_smif.cysmif file | ☐ |

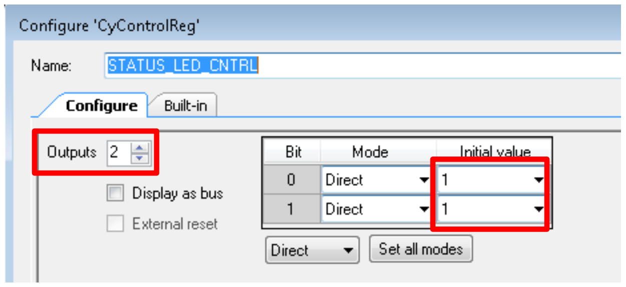
Configure 'SCB_UART_PDL'

Name: UART

Basic | Advanced | Pins | Built-in

| | |
|---|---|
| **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 | ☐ |

Figure 8. Settings for Control Register (Non-default settings are outlined)



Configure 'CyControlReg'

Name: STATUS_LED_CNTRL

Configure | Built-in

Outputs: 2

☐ Display as bus

☐ External reset

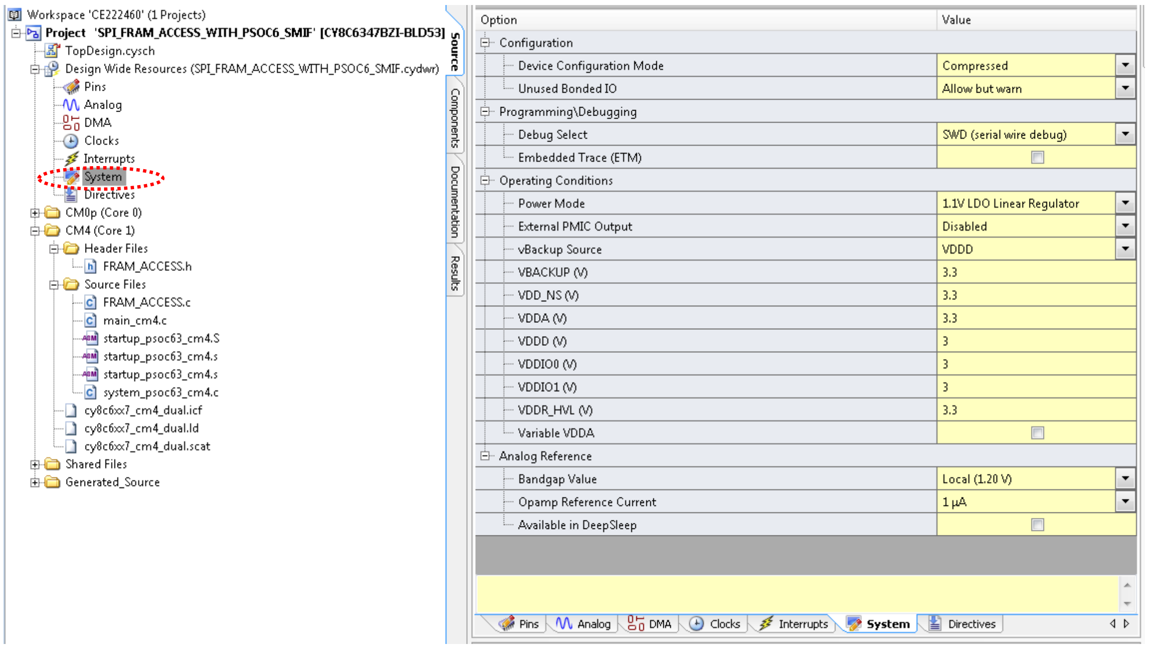
| Bit | Mode | Initial value |
|---|---|---|
| 0 | Direct | 1 |
| 1 | Direct | 1 |

Direct | Set all modes

## Design-Wide Resources

Make sure that $V_{DDD}$ (**PSoC Creator** > **Design Wide Resources** > **System** tab) is set to 2.7 V or above, as shown in Figure 9, to drive the STATUS_LED. Also, make sure that the PSoC 6 MCU I/O voltage is set correctly to match the QSPI F-RAM operating range ($V_{DD}$/ $V_{CC}$).

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

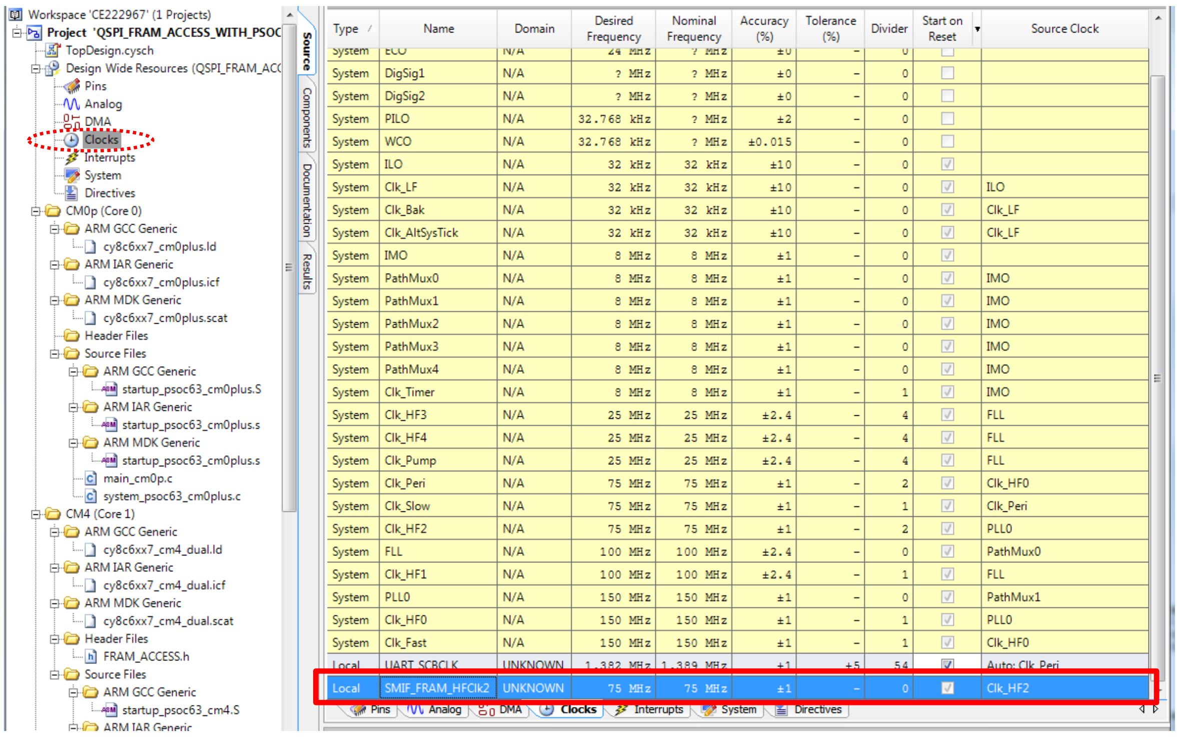


```
Workspace 'CE222460' (1 Projects)
Project 'SPI_FRAM_ACCESS_WITH_PSOC6_SMIF' [CY8C6347BZI-BLD53]
  TopDesign.cysch
  Design Wide Resources (SPI_FRAM_ACCESS_WITH_PSOC6_SMIF.cydwr)
    Pins
    Analog
    DMA
    Clocks
    Interrupts
    System
    Directives
  CM0p (Core 0)
  CM4 (Core 1)
    Header Files
      FRAM_ACCESS.h
    Source Files
      FRAM_ACCESS.c
      main_cm4.c
      startup_psoc63_cm4.S
      startup_psoc63_cm4.s
      startup_psoc63_cm4.s
      system_psoc63_cm4.c
    cy8c6xx7_cm4_dual.icf
    cy8c6xx7_cm4_dual.ld
    cy8c6xx7_cm4_dual.scat
  Shared Files
  Generated_Source
```

| Option | Value |
|---|---|
| Configuration | |
| Device Configuration Mode | Compressed |
| Unused Bonded IO | Allow but warn |
| Programming\Debugging | |
| Debug Select | SWD (serial wire debug) |
| Embedded Trace (ETM) | ☐ |
| Operating Conditions | |
| Power Mode | 1.1V LDO Linear Regulator |
| External PMIC Output | Disabled |
| vBackup Source | VDDD |
| VBACKUP (V) | 3.3 |
| VDD_NS (V) | 3.3 |
| VDDA (V) | 3.3 |
| VDDD (V) | 3 |
| VDDIO0 (V) | 3 |
| VDDIO1 (V) | 3 |
| VDDR_HVL (V) | 3.3 |
| Variable VDDA | ☐ |
| Analog Reference | |
| Bandgap Value | Local (1.20 V) |
| Opamp Reference Current | 1 µA |
| Available in DeepSleep | ☐ |

Make sure that the SMIF clock frequency is set to 80 MHz or below. This code example sets the SMIF_FRAM clock (SMIF_FRAM_HFClk2) to 75 MHz, as shown via Figure 10 and Figure 12. Go to **PSoC Creator** > **Design Wide Resources** and click **Clocks**.

Figure 10. SMIF_FRAM_HFClk2 Setting using Design Wide Resources – Step 1

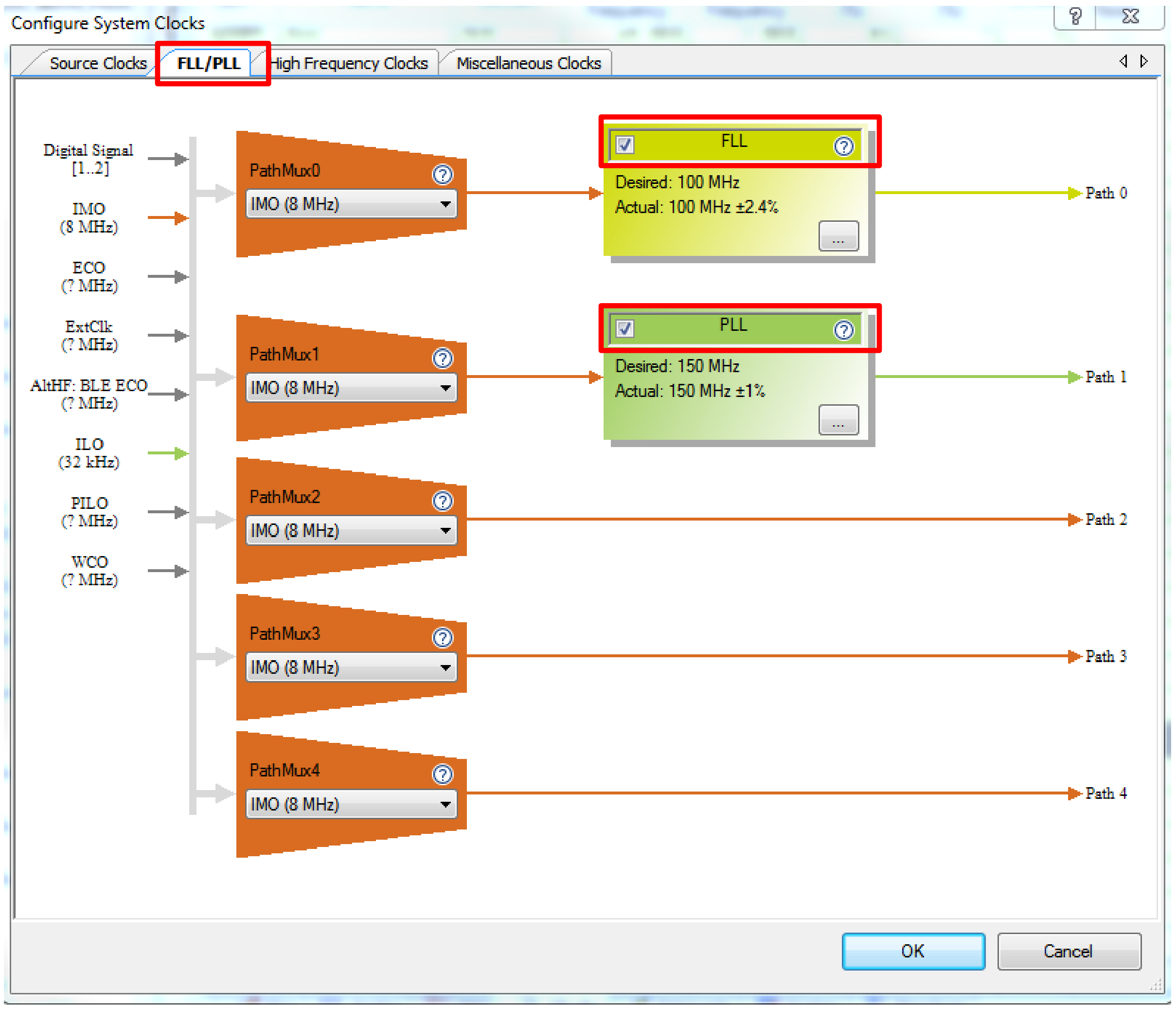


```
Workspace 'CE222967' (1 Projects)
  Project 'QSPI_FRAM_ACCESS_WITH_PSOC
    TopDesign.cysch
    Design Wide Resources (QSPI_FRAM_ACC
      Pins
      Analog
      DMA
      Clocks
      Interrupts
      System
      Directives
    CM0p (Core 0)
      ARM GCC Generic
        cy8c6xx7_cm0plus.ld
      ARM IAR Generic
        cy8c6xx7_cm0plus.icf
      ARM MDK Generic
        cy8c6xx7_cm0plus.scat
      Header Files
      Source Files
        ARM GCC Generic
          startup_psoc63_cm0plus.S
        ARM IAR Generic
          startup_psoc63_cm0plus.s
        ARM MDK Generic
          startup_psoc63_cm0plus.s
        main_cm0p.c
        system_psoc63_cm0plus.c
    CM4 (Core 1)
      ARM GCC Generic
        cy8c6xx7_cm4_dual.ld
      ARM IAR Generic
        cy8c6xx7_cm4_dual.icf
      ARM MDK Generic
        cy8c6xx7_cm4_dual.scat
      Header Files
        FRAM_ACCESS.h
      Source Files
        ARM GCC Generic
          startup_psoc63_cm4.S
        ARM IAR Generic
```

| Type | Name | Domain | Desired Frequency | Nominal Frequency | Accuracy (%) | Tolerance (%) | Divider | Start on Reset | Source Clock |
|---|---|---|---|---|---|---|---|---|---|
| System | ECO | N/A | 24 MHz | ? MHz | ±0 | – | 0 | ☐ | |
| System | DigSig1 | N/A | ? MHz | ? MHz | ±0 | – | 0 | ☐ | |
| System | DigSig2 | N/A | ? MHz | ? MHz | ±0 | – | 0 | ☐ | |
| System | PILO | N/A | 32.768 kHz | ? MHz | ±2 | – | 0 | ☐ | |
| System | WCO | N/A | 32.768 kHz | ? MHz | ±0.015 | – | 0 | ☐ | |
| System | ILO | N/A | 32 kHz | 32 kHz | ±10 | – | 0 | ✓ | |
| System | Clk_LF | N/A | 32 kHz | 32 kHz | ±10 | – | 0 | ✓ | ILO |
| System | Clk_Bak | N/A | 32 kHz | 32 kHz | ±10 | – | 0 | ✓ | Clk_LF |
| System | Clk_AltSysTick | N/A | 32 kHz | 32 kHz | ±10 | – | 0 | ✓ | Clk_LF |
| System | IMO | N/A | 8 MHz | 8 MHz | ±1 | – | 0 | ✓ | |
| System | PathMux0 | N/A | 8 MHz | 8 MHz | ±1 | – | 0 | ✓ | IMO |
| System | PathMux1 | N/A | 8 MHz | 8 MHz | ±1 | – | 0 | ✓ | IMO |
| System | PathMux2 | N/A | 8 MHz | 8 MHz | ±1 | – | 0 | ✓ | IMO |
| System | PathMux3 | N/A | 8 MHz | 8 MHz | ±1 | – | 0 | ✓ | IMO |
| System | PathMux4 | N/A | 8 MHz | 8 MHz | ±1 | – | 0 | ✓ | IMO |
| System | Clk_Timer | N/A | 8 MHz | 8 MHz | ±1 | – | 1 | ✓ | IMO |
| System | Clk_HF3 | N/A | 25 MHz | 25 MHz | ±2.4 | – | 4 | ✓ | FLL |
| System | Clk_HF4 | N/A | 25 MHz | 25 MHz | ±2.4 | – | 4 | ✓ | FLL |
| System | Clk_Pump | N/A | 25 MHz | 25 MHz | ±2.4 | – | 4 | ✓ | FLL |
| System | Clk_Peri | N/A | 75 MHz | 75 MHz | ±1 | – | 2 | ✓ | Clk_HF0 |
| System | Clk_Slow | N/A | 75 MHz | 75 MHz | ±1 | – | 1 | ✓ | Clk_Peri |
| System | Clk_HF2 | N/A | 75 MHz | 75 MHz | ±1 | – | 2 | ✓ | PLL0 |
| System | FLL | N/A | 100 MHz | 100 MHz | ±2.4 | – | 0 | ✓ | PathMux0 |
| System | Clk_HF1 | N/A | 100 MHz | 100 MHz | ±2.4 | – | 1 | ✓ | FLL |
| System | PLL0 | N/A | 150 MHz | 150 MHz | ±1 | – | 0 | ✓ | PathMux1 |
| System | Clk_HF0 | N/A | 150 MHz | 150 MHz | ±1 | – | 1 | ✓ | PLL0 |
| System | Clk_Fast | N/A | 150 MHz | 150 MHz | ±1 | – | 1 | ✓ | Clk_HF0 |
| Local | UART_SCBCLK | UNKNOWN | 1.382 MHz | 1.389 MHz | ±1 | +5 | 54 | ✓ | Auto: Clk_Peri |
| Local | SMIF_FRAM_HFClk2 | UNKNOWN | 75 MHz | 75 MHz | ±1 | – | 0 | ✓ | Clk_HF2 |

Double-click anywhere in the type “system”, highlighted in yellow; for example, **Clk_HF0** or **Clk_Fast** row in Figure 10. A new **Configure System Clock** window opens. Select the **FLL/PLL** tab and check the FLL and PLL clock options with clock frequencies set as 100 MHz for the FLL and 150 MHz for the PLL, as shows in Figure 12.

Figure 11. FLL/PLL Clock Setting – Step 2


Configure System Clocks

Tabs: Source Clocks, FLL/PLL, High Frequency Clocks, Miscellaneous Clocks

Sources: Digital Signal [1..2]; IMO (8 MHz); ECO (? MHz); ExtClk (? MHz); AltHF: BLE ECO (? MHz); ILO (32 kHz); PILO (? MHz); WCO (? MHz)

PathMux0: IMO (8 MHz) → FLL [✓] Desired: 100 MHz Actual: 100 MHz ±2.4% → Path 0

PathMux1: IMO (8 MHz) → PLL [✓] Desired: 150 MHz Actual: 150 MHz ±1% → Path 1

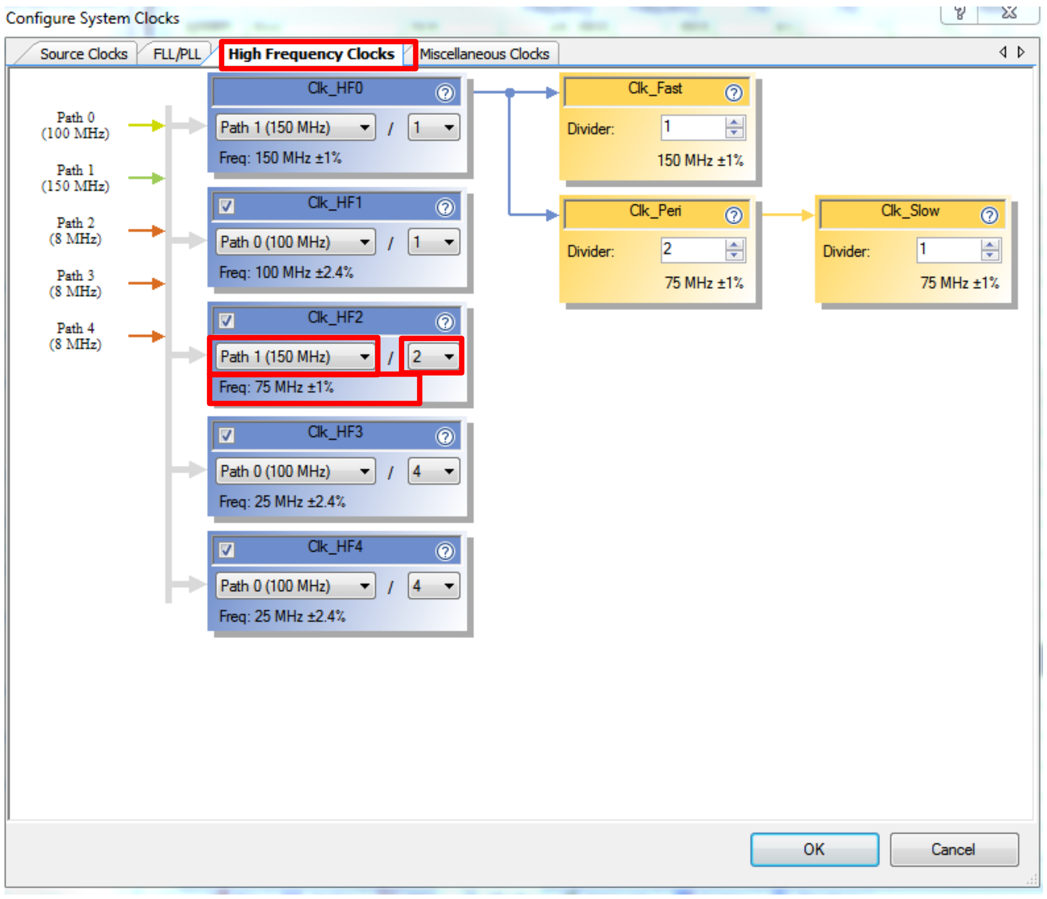
PathMux2: IMO (8 MHz) → Path 2

PathMux3: IMO (8 MHz) → Path 3

PathMux4: IMO (8 MHz) → Path 4

Select the **High Frequency Clocks** tab and select the appropriate clock path and clock divider from their drop-down options, as highlighted Figure 12, to achieve a frequency of 75 MHz. Click **OK**. You can use FLL/PLL tab to configure other frequency options for Path 0. The SMIF block currently supports maximum clock frequency of up to 80 MHz. See the PSoC 6 MCU: PSoC 63 with BLE Datasheet for more details on SMIF block features.

Figure 12. SMIF_FRAM_HFClk2 Clock Frequency Setting – Step 3


Configure System Clocks

Source Clocks | FLL/PLL | **High Frequency Clocks** | Miscellaneous Clocks

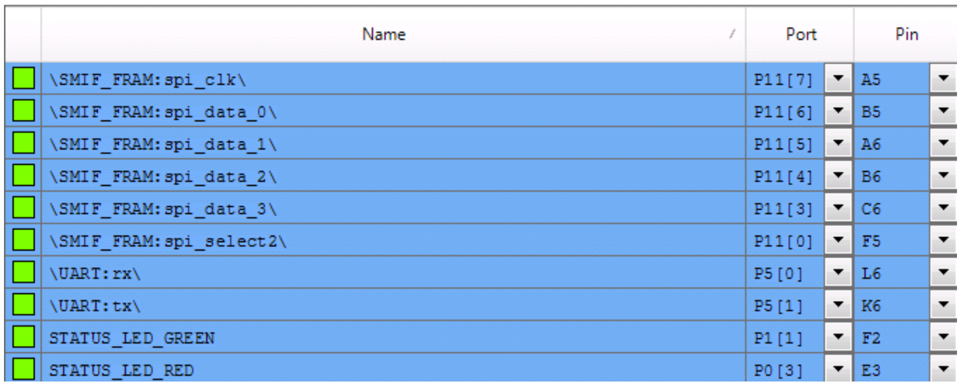
Path 0 (100 MHz)
Path 1 (150 MHz)
Path 2 (8 MHz)
Path 3 (8 MHz)
Path 4 (8 MHz)

Clk_HF0: Path 1 (150 MHz) / 1 — Freq: 150 MHz ±1%
Clk_HF1: Path 0 (100 MHz) / 1 — Freq: 100 MHz ±2.4%
Clk_HF2: Path 1 (150 MHz) / 2 — Freq: 75 MHz ±1%
Clk_HF3: Path 0 (100 MHz) / 4 — Freq: 25 MHz ±2.4%
Clk_HF4: Path 0 (100 MHz) / 4 — Freq: 25 MHz ±2.4%

Clk_Fast: Divider: 1 — 150 MHz ±1%
Clk_Peri: Divider: 2 — 75 MHz ±1%
Clk_Slow: Divider: 1 — 75 MHz ±1%

Figure 13 shows the pin assignment for the code example.

Figure 13. PSoC 6 Pin Assignments for Code Example



| | Name | Port | Pin |
|---|---|---|---|
| 🟩 | \SMIF_FRAM:spi_clk\ | P11[7] | A5 |
| 🟩 | \SMIF_FRAM:spi_data_0\ | P11[6] | B5 |
| 🟩 | \SMIF_FRAM:spi_data_1\ | P11[5] | A6 |
| 🟩 | \SMIF_FRAM:spi_data_2\ | P11[4] | B6 |
| 🟩 | \SMIF_FRAM:spi_data_3\ | P11[3] | C6 |
| 🟩 | \SMIF_FRAM:spi_select2\ | P11[0] | F5 |
| 🟩 | \UART:rx\ | P5[0] | L6 |
| 🟩 | \UART:tx\ | P5[1] | K6 |
| 🟩 | STATUS_LED_GREEN | P1[1] | F2 |
| 🟩 | STATUS_LED_RED | P0[3] | E3 |

## Project Folder and Files Details

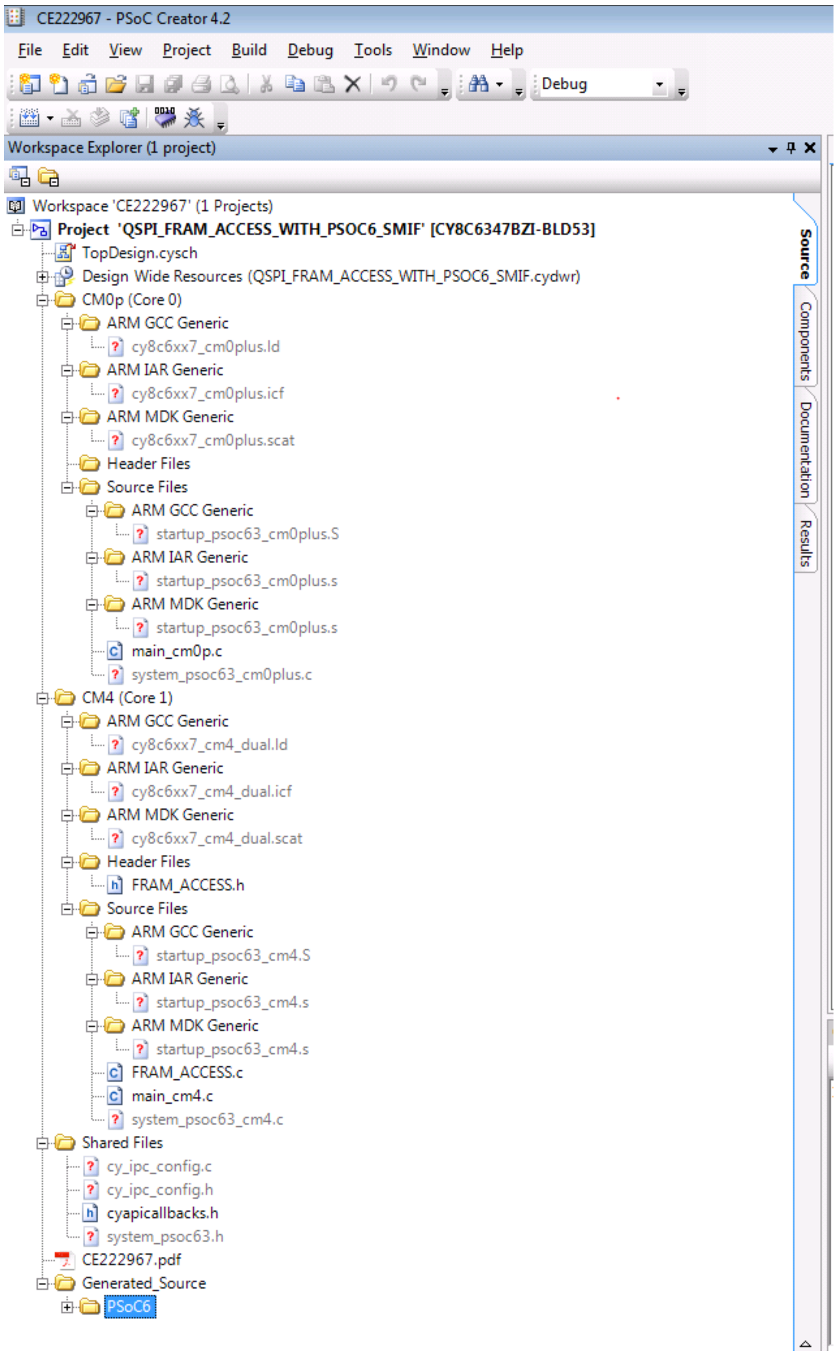
Figure 14 shows the project folder and files structure. The project files mark with '?' are the files missing in the code example provided. These files are automatically compiled and added to the project after a successful build.

- *FRAM_ACCESS.h* – This file declares the function prototype and constants for the QSPI F-RAM access.
- *FRAM_ACCESS.c* – This file defines user functions and APIs for the QSPI F-RAM access. Functions and APIs defined in this file use low-level driver, defined in the smif driver file *cy_smif.c*. The *cy_smif.c* file is in the workspace:

**Project (QSPI_FRAM_ACCESS_WITH_PSOC6_SMIF)** > **Generated_Source** > **PSoC6** > **pdl > drivers** >**peripheral**> **smif**

*main_cm4.c* – This is the *main.c* file where a few APIs are called to demonstrate accessing the QSPI F-RAM using SMIF.

Figure 14. Folder and File Structure for Code Example


CE222967 - PSoC Creator 4.2
File Edit View Project Build Debug Tools Window Help
Debug
Workspace Explorer (1 project)
Source
Components
Documentation
Results

```
Workspace 'CE222967' (1 Projects)
  Project 'QSPI_FRAM_ACCESS_WITH_PSOC6_SMIF' [CY8C6347BZI-BLD53]
    TopDesign.cysch
    Design Wide Resources (QSPI_FRAM_ACCESS_WITH_PSOC6_SMIF.cydwr)
    CM0p (Core 0)
      ARM GCC Generic
        cy8c6xx7_cm0plus.ld
      ARM IAR Generic
        cy8c6xx7_cm0plus.icf
      ARM MDK Generic
        cy8c6xx7_cm0plus.scat
      Header Files
      Source Files
        ARM GCC Generic
          startup_psoc63_cm0plus.S
        ARM IAR Generic
          startup_psoc63_cm0plus.s
        ARM MDK Generic
          startup_psoc63_cm0plus.s
        main_cm0p.c
        system_psoc63_cm0plus.c
    CM4 (Core 1)
      ARM GCC Generic
        cy8c6xx7_cm4_dual.ld
      ARM IAR Generic
        cy8c6xx7_cm4_dual.icf
      ARM MDK Generic
        cy8c6xx7_cm4_dual.scat
      Header Files
        FRAM_ACCESS.h
      Source Files
        ARM GCC Generic
          startup_psoc63_cm4.S
        ARM IAR Generic
          startup_psoc63_cm4.s
        ARM MDK Generic
          startup_psoc63_cm4.s
        FRAM_ACCESS.c
        main_cm4.c
        system_psoc63_cm4.c
    Shared Files
      cy_ipc_config.c
      cy_ipc_config.h
      cyapicallbacks.h
      system_psoc63.h
    CE222967.pdf
    Generated_Source
      PSoC6
```

## Reusing This Example

This example is designed for the CY8CKIT-062-WiFi-BT Pioneer Kit with serial F-RAM mounted on it. To port the design to a different PSoC 6 MCU device, kit, or both, change the target device using **Device Selector** and update the pin assignments in **Design Wide Resources Pins** settings. For single-CPU PSoC 6 MCU devices, port the code from *main_cm4.c* to *main.c*.

## Related Documents

| Application Notes/Code Examples | |
|---|---|
| CE220823 – PSoC 6 MCU SMIF Memory Write and Read Operation | This example demonstrates the write and read operations to the Serial Memory Interface (SMIF) in PSoC 6 MCU. |
| Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity | This application note helps you explore the PSoC 6 MCU with BLE architecture and development tools and shows you how to create your first project using PSoC Creator, export the project to a third-party integrated development environment (IDE), and continue your firmware development. |
| **PSoC Creator Component Datasheets** | |
| UART | UART communications interface |
| Serial Memory Interface (SMIF) | Serial Memory Interface |
| Control Register | Allows the firmware to set values for to use for digital signals |
| General-Purpose Input / Output | Supports Analog, Digital I/O and Bidirectional signal types |
| **Device Documentation** | |
| PSoC 6 MCU: PSoC 63 with BLE Datasheet | PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual |
| QSPI F-RAM (CY15B104QSN) Datasheet | 1.8 V to 3.6 V (3 V typical), 108 MHz QSPI FRAM datasheet |
| **Development Kit (DVK) Documentation** | |
| PSoC 6 WiFi-BT Pioneer Kit (CY8CKIT-062-WiFi-BT) | |

## Document History

Document Title: CE222967 – Excelon-Ultra QSPI F-RAM Access Using PSoC 6 MCU SMIF

Document Number: 002-22967

| Revision | ECN | Submission Date | Description of Change |
|---|---|---|---|
| ** | 6073046 | 02/28/2018 | New Code Example |
| *A | 6282523 | 8/15/2018 | 1. Updated the “PowerUpMemoryDefaultSPI” function in main.c using the “write any register (WRAR)” opcode to write to configuration registers<br>2. Updated the copyright notice in the main.c, FRAM_ACCESS.c, and FRAM_ACCESS.h files<br>3. Updated CONFIG_REG2_ADDR initialization in Global variables and functions section in main.c. Changed the configuration register 2 (CR2) address to volatile register address 0x700003 |
| *B | 6629848 | 07/15/2019 | Fixed the broken hyperlink for the PDL, SMIF, and QSPI FRAM product page link<br>Updated the Related Hardware to PSoC 6 WiFi-BT Pioneer Kit |

## 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