CE224285 – PSoC 6 MCU External Flash Access in XIP Mode with QSPI

Objective

This example demonstrates how to use QSPI in execute-in-place (XIP) mode with external flash memory on the PSoC® 6 MCU using ModusToolbox™ IDE.

Overview

This example uses QSPI in XIP mode to access variables and functions which are stored in an external flash memory device. A UART resource displays the output and status of the program as it executes.

Requirements

Tool: ModusToolbox IDE 1.1

Programming Language: C

Associated Parts: All PSoC 6 MCU parts with QSPI

Related Hardware: PSoC 6 BLE Pioneer Kit, PSoC 6 WiFi-BT Pioneer Kit, PSoC 6 Wi-Fi BT Prototyping Kit

Hardware Setup

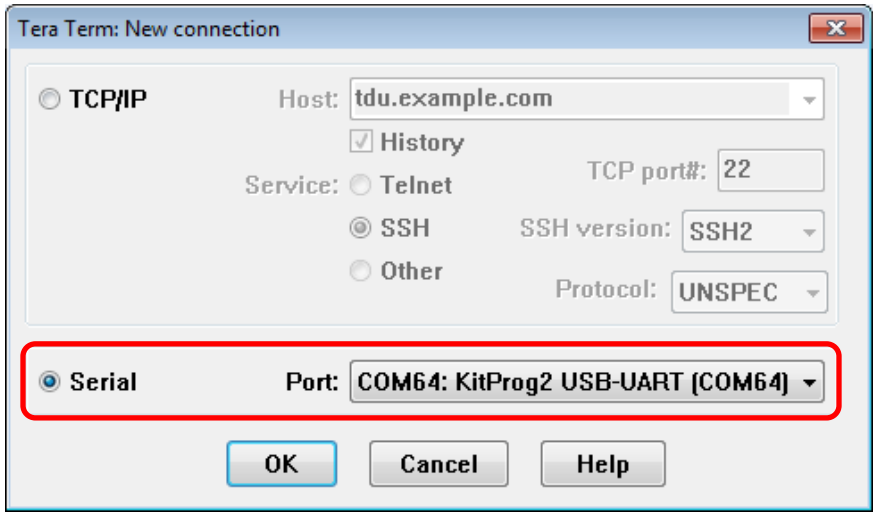
This example uses the kit's default configuration. See the kit guide to ensure that the kit is configured correctly.

Software Setup

This section describes the procedure to set up a serial (UART) connection using Tera Term 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 terminal emulator programs, such as PuTTY, can also be used.

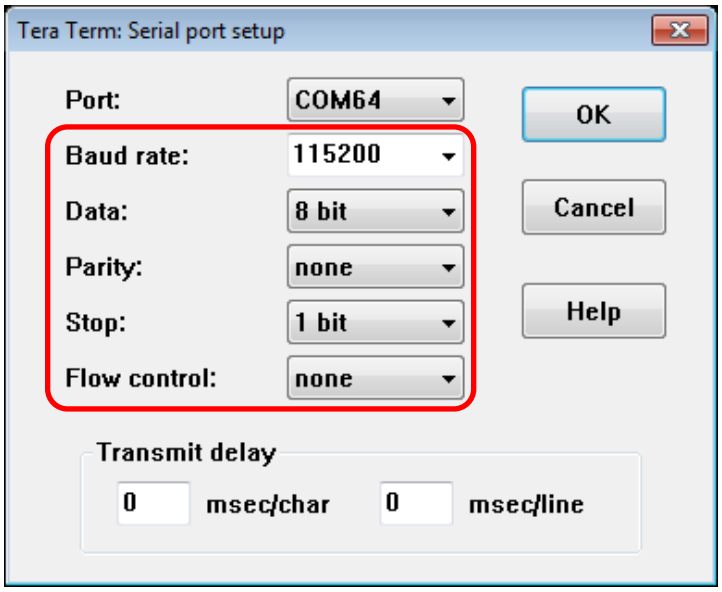
1. Connect header J10 on the PSoC 6 MCU Pioneer Kit to the PC using the USB cable.
2. After installing Tera Term, open the program and select the KitProg2 device from the **Port** dropdown list. Click **OK**.

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



PSoC Programmer™ is optionally used to read the data stored in the external memory. It can be downloaded here.

Operation

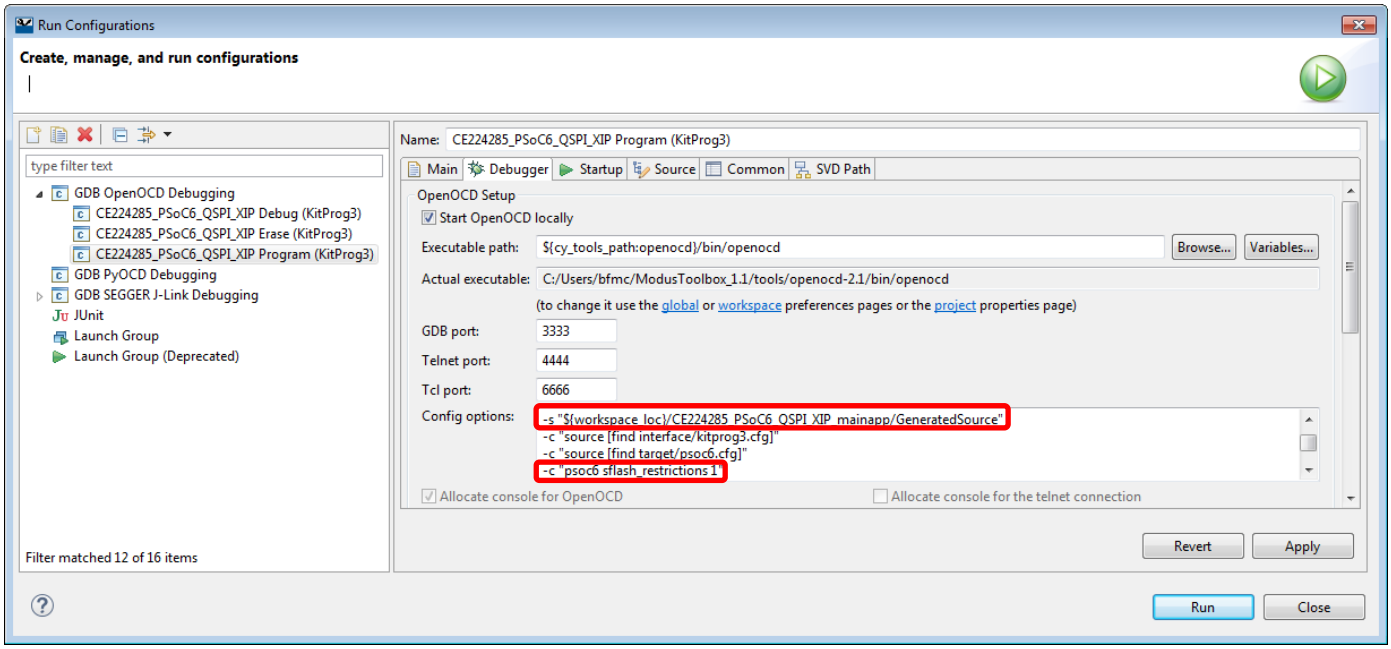
1. Connect the kit and configure the terminal following the instructions in Software Setup.
2. Import the application projects into a new workspace. See KBA225201.
3. Copy the following and paste in line 376 of the linker descriptor file *cy8c6xx7_cm4_dual.ld*.

```
.cy_xip_code :
{
    KEEP(*(.cy_xip))
} > xip
```

4. Right-click the "CE224285_PSoC6_QSPI_XIP_mainapp" project in the Project Explorer. Select **Run As** > **Run Configurations**. In the Run Configurations window, select **GDB OpenOCD Debugging** > **CE224285_PSoC6_QSPI_XIP Program (KitProg3)**. Select the **Debugger** window and add the following options to the **Config options**:
   a. Add `-s "${workspace_loc}/CE224285_PSoC6_QSPI_XIP_mainapp/GeneratedSource"`
   b. After the line `-c "source [find target/psoc6.cfg]"` add `-c "psoc6 sflash_restrictions 1"`

The result should appear as shown in Figure 3.

Figure 3. OpenOCD Run Configurations



5. Build the application. Select the *CE224285_PSoC6_QSPI_XIP_mainapp* project in the Project Explorer. In the ModusToolbox quick panel on the lower left corner of the IDE, select **Build CE224285_PSoC6_QSPI_XIP Application**.
6. Program the application into the PSoC 6 MCU. In the ModusToolbox quick panel, select **CE224285_PSoC6_QSPI_XIP Program (KitProg 3)**.
7. Observe the output of the program on the UART terminal. Confirm that it shows the arrays of data read from the external memory.
8. Open PSoC Programmer and connect to the KitProg device associated with your PSoC 6 MCU Kit.
9. In PSoC Programmer, navigate to the **Memory Types** window on the right side of the application. Select **Load from device** and check the **SMIF** checkbox. Uncheck all other memory types.
10. Select **File** > **Read To Log** and observe the output. A block of hexadecimal values will appear in the Results window. These values are the values stored in the memory-mapped sectors of external memory. Confirm that the first five rows are non-zero values.

Design and Implementation

In XIP mode, the QSPI block maps bus accesses to external memory device addresses. This allows functions and data programmed into an external memory device to be used as if they were in internal memory. In this example a function and an array are both programmed into the external memory device. Typically, a compiler will throw a section type conflict if a variable and function are placed in the same section. To resolve this, another XIP section called ".cy_xip_code" is created in the linker file. The function is placed in this section while the array is placed in the ".cy_xip" section. The firmware also uses the pragma `long_calls` to indicate that the function may be far away from the call site.

On device startup, the SMIF Component is initialized in normal mode and performs reads from and writes to the external memory. Firmware checks the data read from the external memory and compares it with the written data. If the data matches, then the firmware sets the SMIF Component into XIP mode.

Once in XIP mode, a string from external memory is printed to the UART terminal. The function `printFromExternalMemory()` is accessible in XIP mode, which enables the user to print a string to the UART.

SMIF Source Files

The SMIF Configuration tool and the PDL for the SMIF block provide the functions and structures required to access the external memory device. Each of the files is listed below.

- *cy_smif.c* – PDL-provided file that contains the functions needed to set up the SMIF block.
- *cy_smif.h* – PDL-provided header file that contains the inline functions, enumerated types, structures, and function declarations for use with the SMIF block configuration functions.
- *cy_smif_memslot.c* – PDL-provided file that contains the low-level functions necessary to set up and access the external memory device.
- *cy_smif_memslot.h* – PDL-provided file that contains the macros, structure, and function declarations for use with the external memory device configuration functions.
- *cycfg_qspi_memslot.c* – QSPI Configurator-generated file that contains the populated structures that define the operational modes and parameters for use with the *cy_smif_memslot.c* functions. In this example, this file is placed in the project's *Source* directory so that the project can be used without needing to generate the file. For instructions on how this file can be generated, see Appendix A.
- *cycfg_qspi_memslot.h* – QSPI Configurator-generated file that contains the declarations of the structures that define the operational modes and parameters for the external memory device. In this example, this file is placed in the project's *Source* directory so that the project can be used without needing to generate the file. For instructions on how this file can be generated, see Appendix A.

This example uses additional custom files *smif_mem.c* and *smif_mem.h* to access the external flash memory. The files provide high-level wrappers around the functions in *cy_smif_memslot.c* to simplify external memory accesses.

Resources and Settings

Table 1 lists the ModusToolbox resources used in this example, and how they are used in the design. For pin usage and configuration, open the **Pins** tab of the *design.modus* file.

Table 1. Modus Toolbox Resources

Resource	Alias	Purpose
SMIF	SMIF	Enables communication with external memory
UART	UART	Enables visual display of information
Digital Output Pin	KIT_LED1	Enables the use of LED
Digital Input Pin	KIT_BTN1	Enables the use of the button

Figure 4 highlights the non-default settings for SMIF. Figure 5 highlights the non-default settings for the UART. Figure 6 highlights the system settings used in this code example.

Figure 4. The Configuration Settings for SMIF

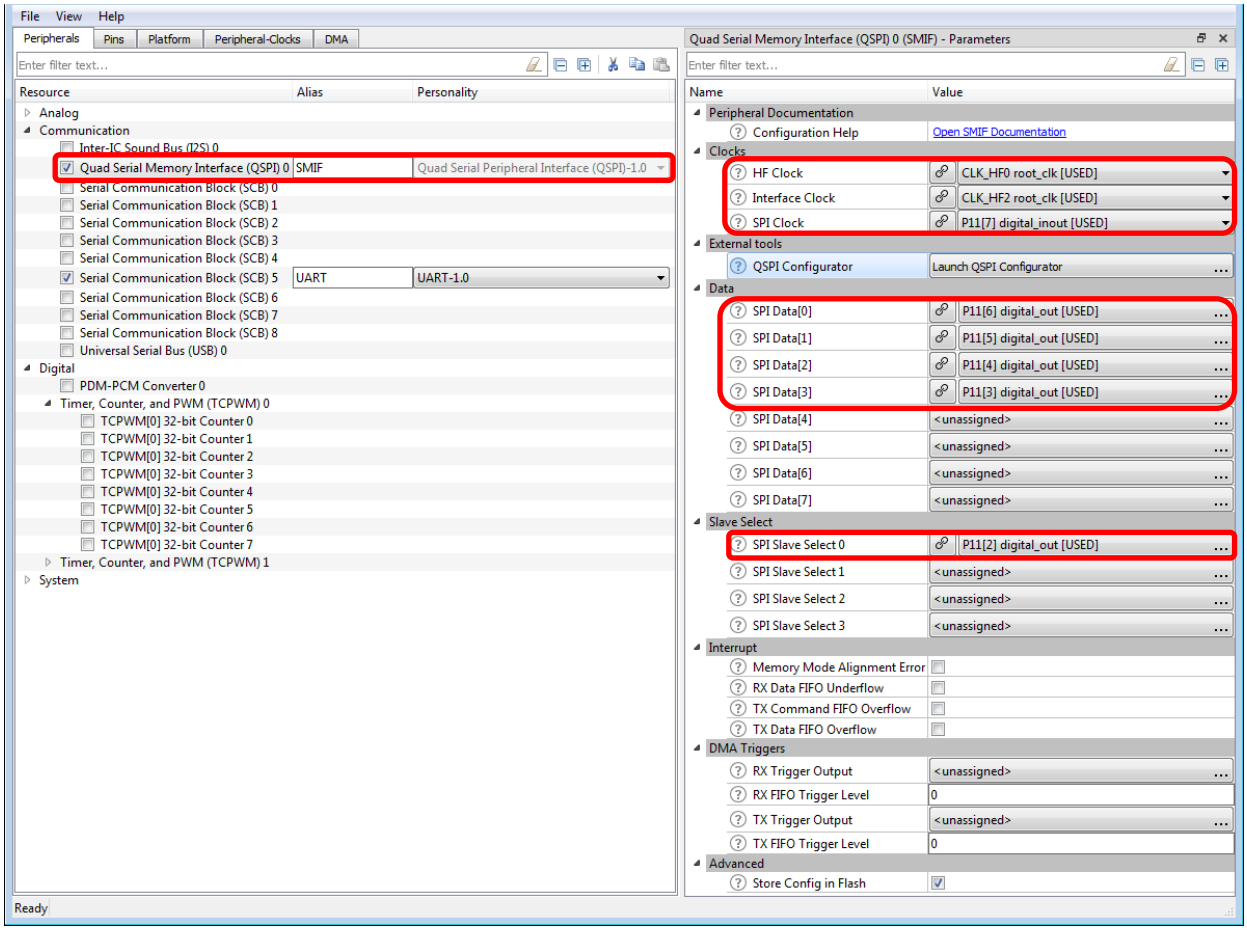


Figure 5. The Configuration Settings for UART

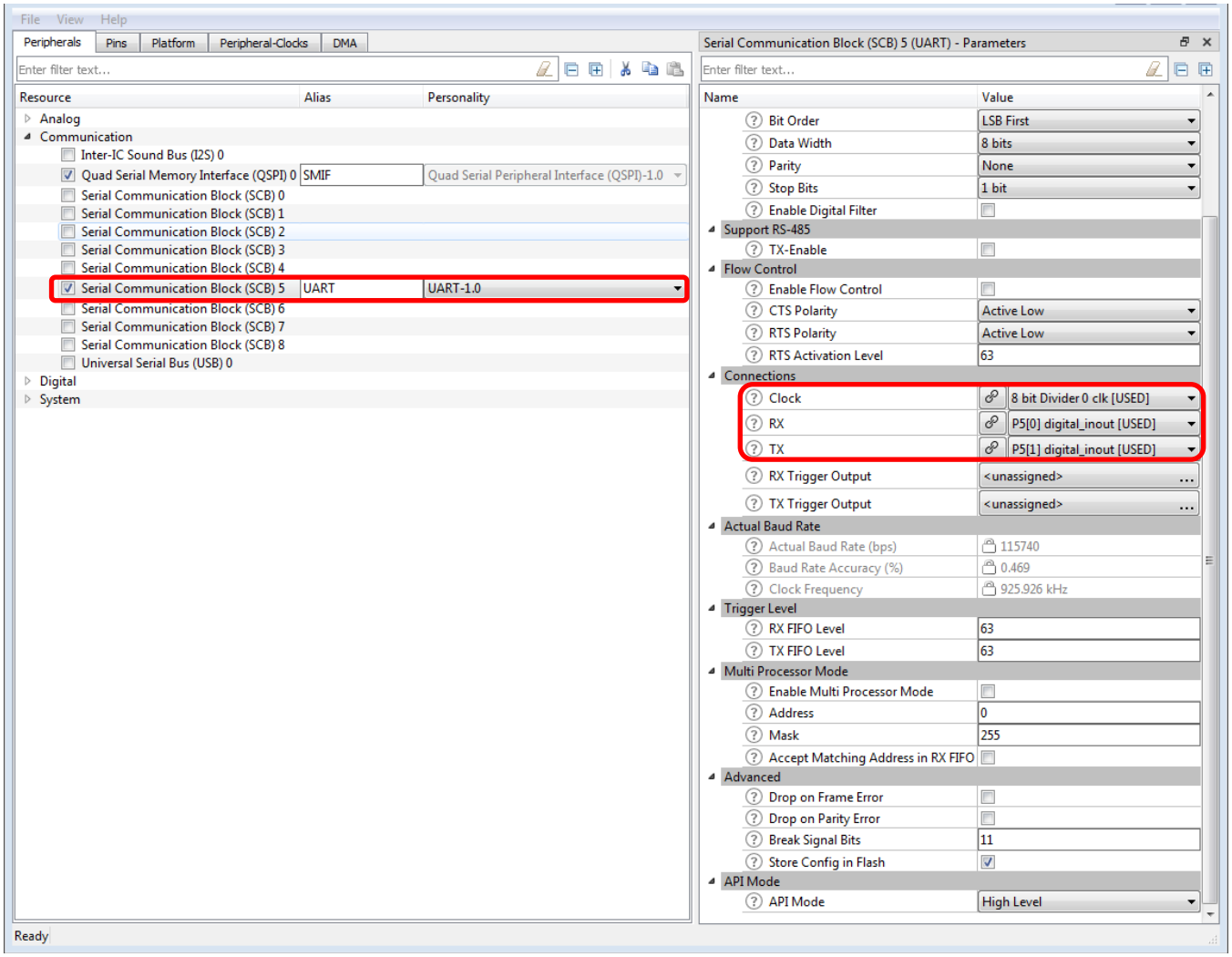
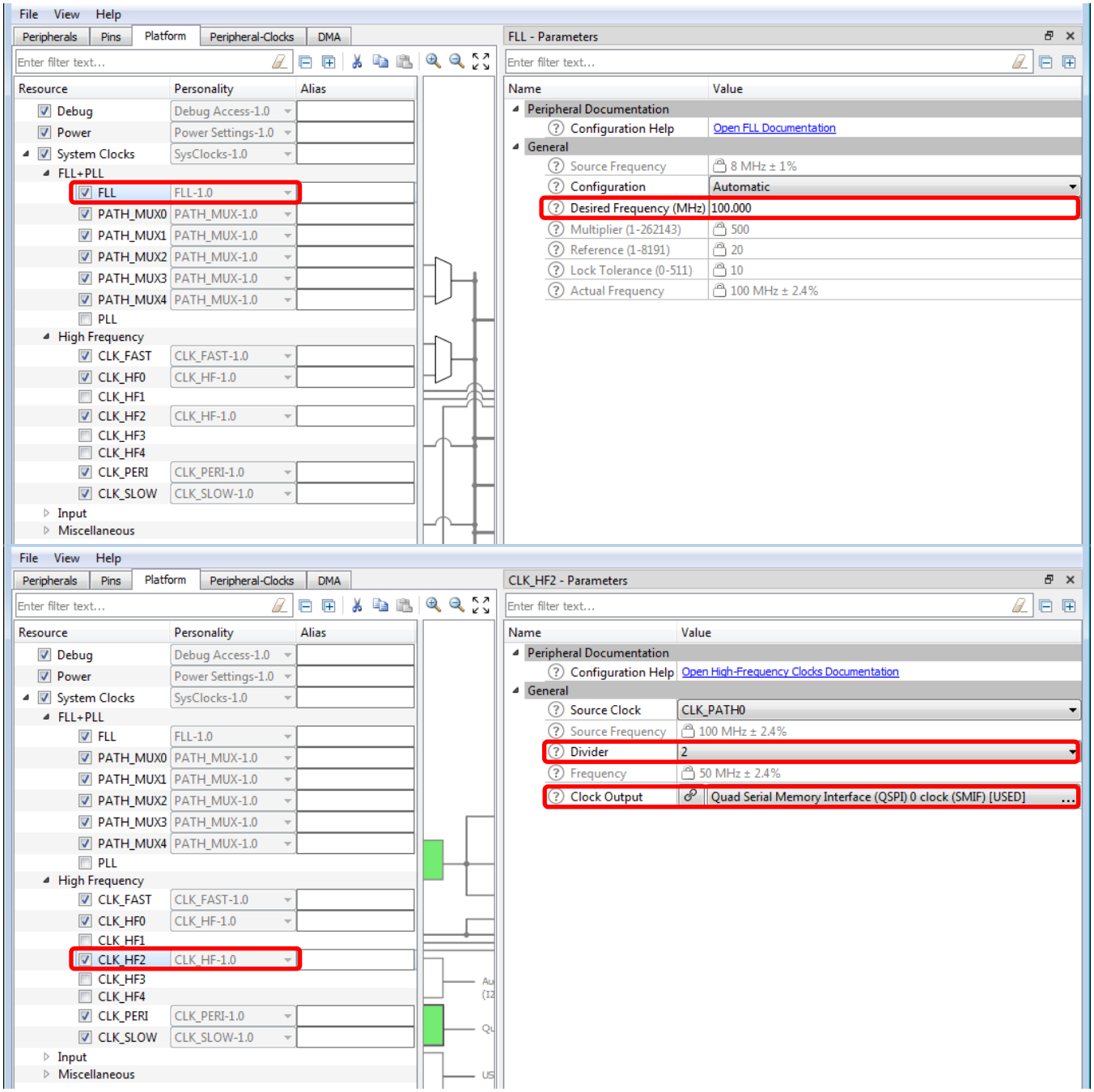


Figure 6. Platform FLL and CLK_HF2 Settings



Reusing This Example

This example is designed for the CY8CKIT-062-WIFI-BT Pioneer Kit. To port the design to a different PSoC 6 MCU device, right click an application project and choose Change Device. If changing to a different kit, you may need to reassign pins.

In some cases, a resource used by a code example is not supported on another device. In that case the example will not work. If you build the code targeted at such a device, you will get errors. See the device datasheet for information on what a particular device supports.

Related Documents

Application Notes	
AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes PSoC 6 MCU with BLE Connectivity devices and how to build your first PSoC Creator project
AN215656 – PSoC 6 MCU: Dual CPU System Design	Describes the dual CPU architecture in PSoC 6 MCU, and shows how to build a simple dual-core design
AN219434 – Importing PSoC Creator Code into an IDE for a PSoC 6 MCU Project	Describes how to import the code generated by PSoC Creator into your preferred IDE
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.
CE222460 – SPI F-RAM Access Using PSoC 6 MCU SMIF	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.
CE224073 – SPI F-RAM Access Using PSoC 6 MCU SMIF in Memory Mapped (XIP) Mode	SPI F-RAM Access Using PSoC 6 MCU SMIF in Memory Mapped (XIP) Mode
PSoC Creator Component Datasheets	
Pins	Supports connection of hardware resources to physical pins
SMIF	Supports external memory access
UART	UART communications interface
Device Documentation	
PSoC® 6 MCU Datasheets	PSoC® 6 MCU Technical Reference Manual
Serial NOR Flash (S25FL512S) Datasheet	
Development Kit Documentation	
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit	
CY8CKIT-062-WiFi-BT PSoC 6 WiFi-BT Pioneer Kit	

Appendix A: SMIF Configuration Tool

Modus Toolbox supports a stand-alone application, QSPI Configurator, which enables you to configure the SMIF through a GUI-based interface. This application is invoked from the SMIF Parameter editor in the *design.modus* file. Figure 7 and Figure 8 show how to configure the memory device interfaced with SMIF. Follow these steps to generate SMIF driver memory configuration (*cycfg_qspi_memslot.c,* and *cycfg_qspi_memslot.h*) files from the SMIF Configuration tool.

1. Remove *cycfg_qspi_memslot.c* and *cycfg_qspi_memslot.h* files from the project workspace.
2. Open the *design.modus* file in ModusToolbox and click **Peripherals**.
3. Under **Resources** > **Communication**, select the **Quad Serial Memory Interface (QSPI)**.
4. In the parameters window, select **QSPI Configurator**.
5. Configure the memory part number to match the device on the kit. In this case, **S25FL512S**.
6. Click **File** > **Save Configuration**.
7. Click **Options** > **Configurations…** to select the *example* root folder as the output folder.
8. Close the QSPI Configurator tool.
9. Build the application. Modus Toolbox generates the *cycfg_qspi_memslot.c* and *cycfg_qspi_memslot.h* files. These files provide the external memory parameters for use with the functions in *cy_smif_memslot.c*.

Figure 7. SMIF Configuration Tool

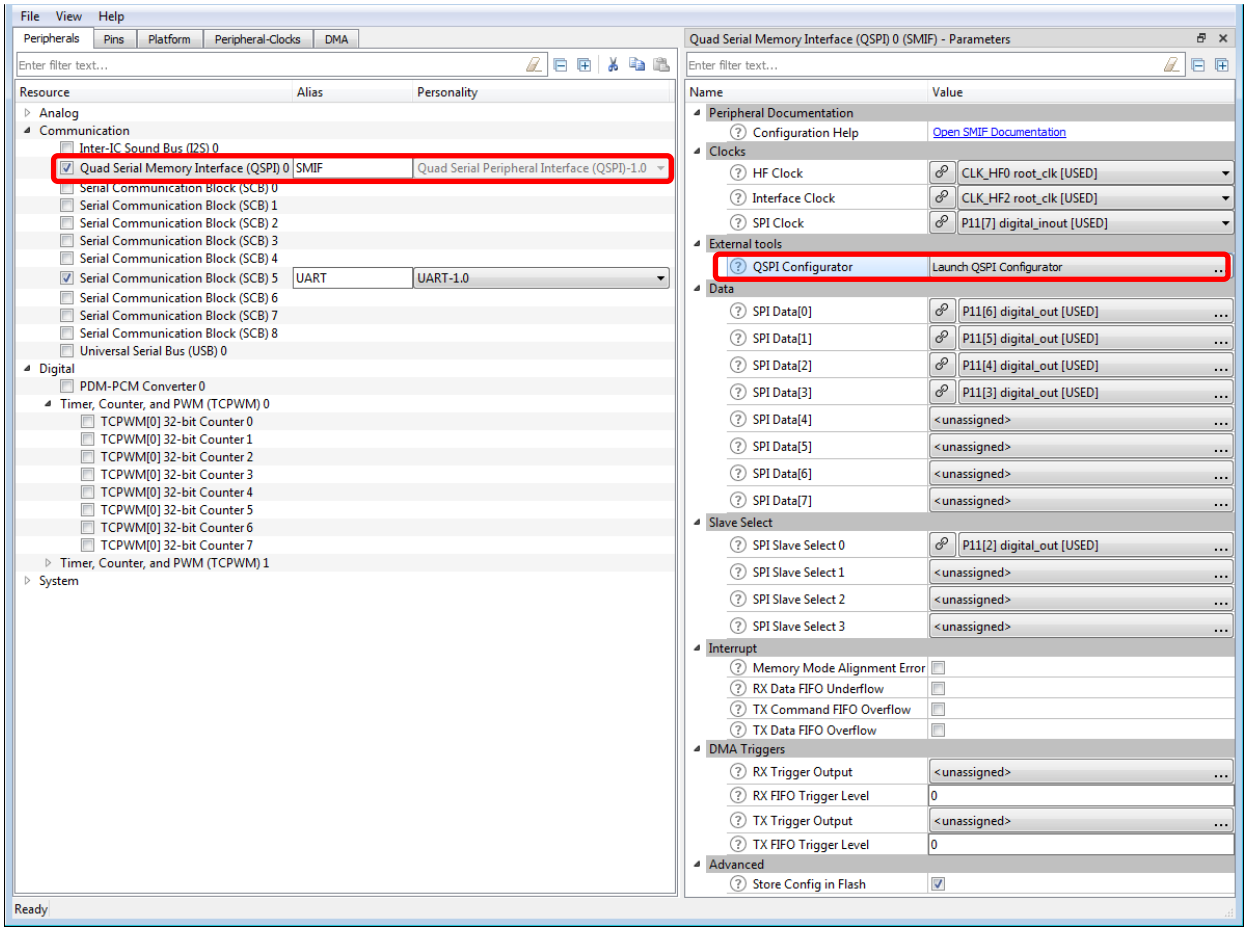
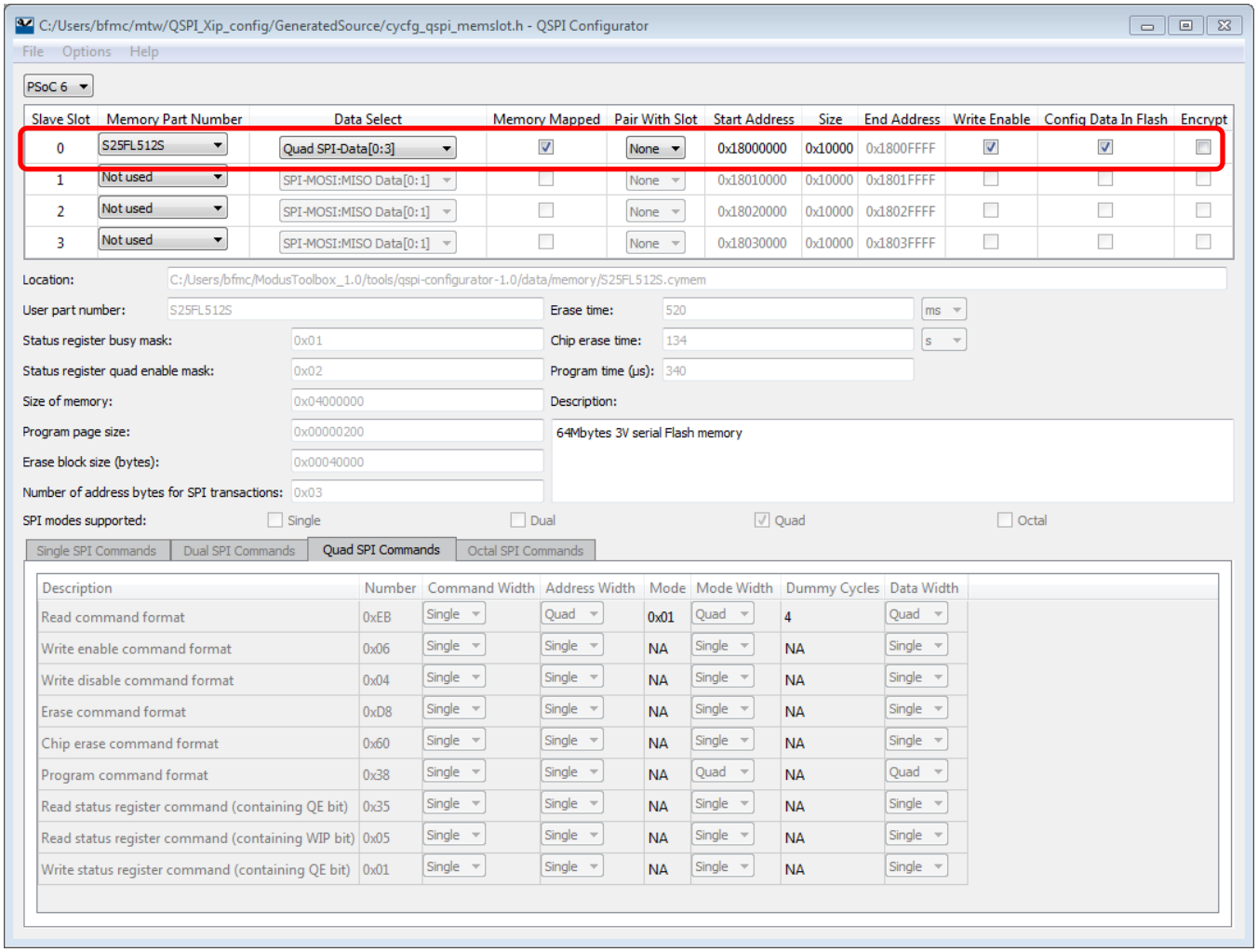


Figure 8. SMIF Configuration Tool Memory Configuration



The SMIF Configuration tool can also be used for external memories which are not listed in the memory part number dropdown list. To create a custom memory number, follow these steps.

1. Remove *cycfg_qspi_memconfig.c* and *cycfg_qspi_memconfig.h* files from the project workspace.
2. Open the *design.modus* device configurator and navigate to **Peripherals** > **Quad Serial Memory Interface**.
3. Select **Launch QSPI Configurator**. A new window, 'Edit Memory', opens.
4. In the **Edit Memory** window, select **File** > **New *.cymem File**. Edit the location to store the new *.cymem* file. Navigate to *C:\Program Files (x86)\Cypress\PDL\3.0.1\tools\win\smif_config\memory.*

Other directories can be used; however, the device will not appear in the memory part number dropdown menu. Instead, the custom *.cymem* file must be selected using the **<browse…>** option at the bottom of the memory part number dropdown list.

1. Enter the custom name for the part in the **User part number** field.
2. Fill in the remaining fields following information from the device datasheet.
3. Select **Save** and follow steps 4-9 of the previous section.

Document History

Document Title: CE224285 – PSoC 6 MCU External Flash Access in XIP Mode with QSPI

Document Number: 002-26382

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6557757	BFMC	06/04/2019	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	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 | Code Examples | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

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