Supporting a new flash device using QSPI Configurator tool in ModusToolbox™ for PSOC™ Edge MCUs

About this document

Scope and purpose

This application note provides comprehensive information on how to generate a `.cymem` file for flash parts that are not listed in QSPI Configurator and use it effectively.

Intended audience

This document is intended for users who want to use additional flash parts that are not listed in QSPI Configurator in ModusToolbox™.

Table of contents

About this document . . . 1
Table of contents . . . 2
1 Introduction . . . 3
1.1 QSPI Configurator . . . 3
1.2 QSPI Configurator features . . . 3
2 Adding unlisted external memory using CYMEM file . . . 4
2.1 Creating a CYMEM file . . . 4
2.1.1 Adding a non HYPERBUS™ device (SEMPER™ NOR flash) . . . 5
2.1.2 Adding a HYPERBUS™ device . . . 13
2.2 Using the CYMEM file . . . 14
2.3 Fine-tuning the CYMEM file . . . 16
2.4 SFDP memories supported in PSOC™ Edge MCU . . . 18
3 Implementing DDR using NOR flash . . . 20
3.1 Constraints and recommendations of using Octal SPI in DDR mode for PSOC™ Edge MCU . . . 21
4 QSPI vs. OSPI performance comparison . . . 22
4.1 Use cases and selection guidance . . . 24
4.1.1 QSPI NOR flash . . . 24
4.1.2 OSPI NOR flash . . . 24
4.1.3 OSPI PSRAM . . . 25
5 Exclusions . . . 26
References . . . 27
Revision history . . . 28
Disclaimer . . . 29

1 Introduction

1.1 QSPI Configurator

PSOC™ Edge MCUs include internal SRAM and non-volatile flash memory. Applications that require additional storage or execution memory can connect external memory devices through the Serial Memory Interface (SMIF).

ModusToolbox™ provides Serial Memory Interface (SMIF) drivers that help in interacting with the external memory/flash part. The SMIF driver provides APIs to read/write/erase memory. You can configure memory parts using the QSPI Configurator and generate source files containing the commands, structures, and other information about the memory part that can be used with the SMIF driver.

1.2 QSPI Configurator features

QSPI Configurator can be used to configure memory devices in various modes, such as single SPI, Dual SPI, Quad SPI, or Octal SPI mode.

QSPI Configurator supports various memory devices and they can be selected from the drop down under **Memory Part Number**.

If the external memory supports the Serial Flash Discoverable Parameter (SFDP) standard, the SMIF drivers can query the memories connected to the PSOC™ Edge device.

If SFDP is supported by the external memory, **Auto Detect SFDP** can be selected from the **Memory Part Number**.

QSPI Configurator allows you to make other configuration changes, such as the number for slave select lines to be used, encryption options, memory mapping with device, etc. Once the required configurations are implemented and saved, code is generated and placed in the `Generated Source` folder in the BSP.

For more details on SMIF driver and QSPI Configurator, refer SMIF driver documentation and QSPI Configurator user guide.

2 Adding unlisted external memory using CYMEM file

As mentioned in the previous section, the QSPI Configurator supports a large number of flash parts and they are listed in the drop down under **Memory Part Number** in QSPI Configurator.

To use a memory part that is not listed, a CYMEM file must be created.

2.1 Creating a CYMEM file

To create a CYMEM file, follow these steps:

1. Launch QSPI Configurator. Click **File** > **New *.cymem File...** as shown in Figure 1

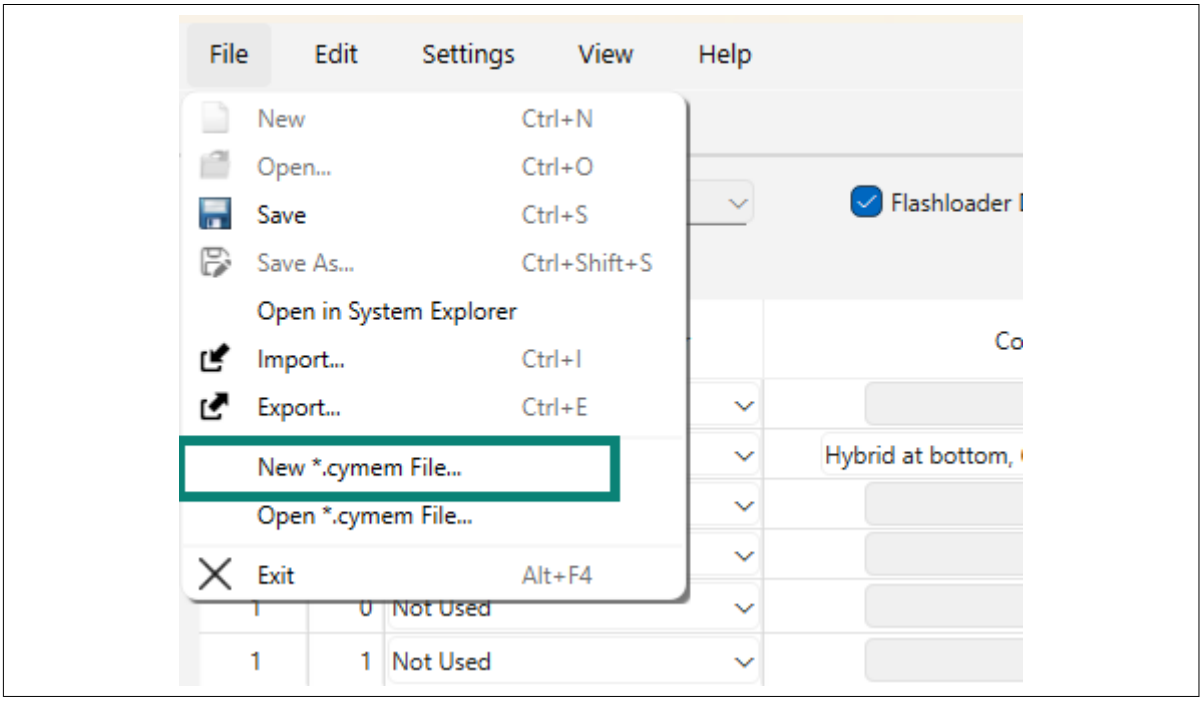

Figure 1 Create a new CYMEM file

2. The **Edit Memory** window opens, which can be used to add a flash/memory device, such as SEMPER™ NOR flash to QSPI Configurator and can also be used to add a new HYPERBUS™ device

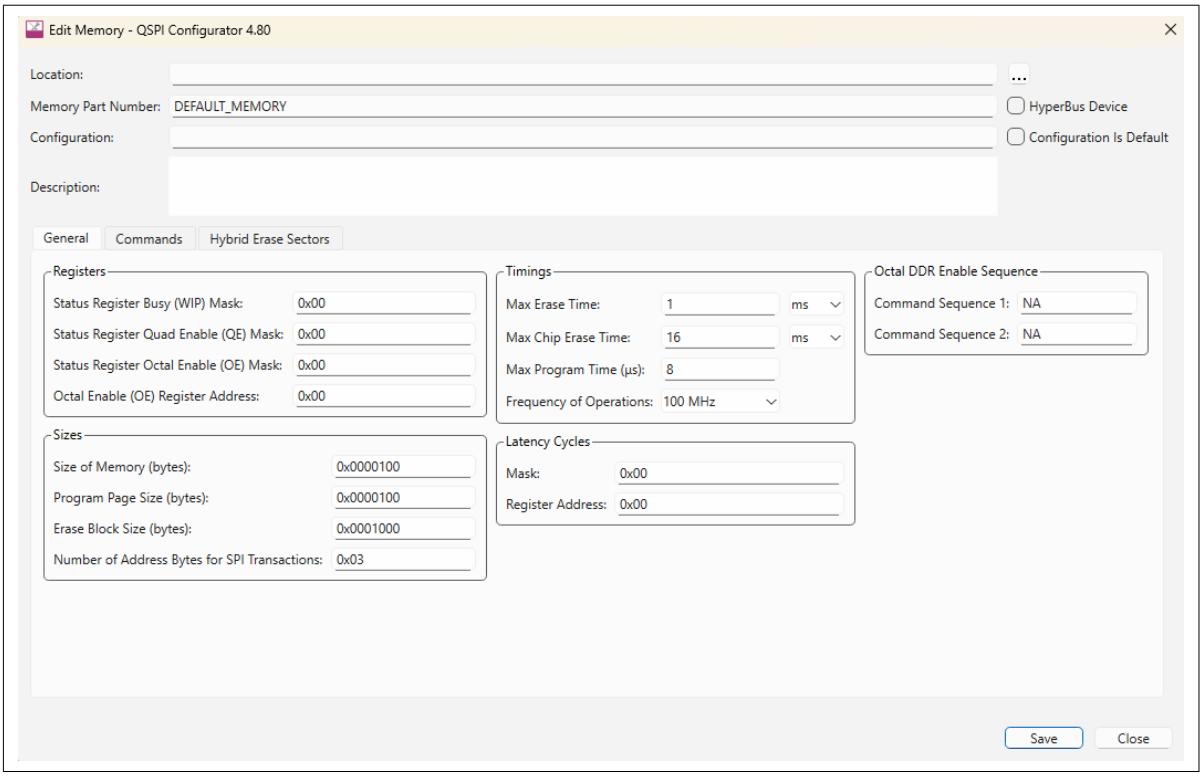

Figure 2 Edit Memory

2.1.1 Adding a non HYPERBUS™ device (SEMPER™ NOR flash)

1. **Location**: Click on the **...** button to specify location to save the `CYMEM` file
2. **Memory Part Number**: This is the name shown in the drop down in QSPI Configurator
3. **Configuration**: If the memory part can be operated in multiple sector architectures or modes, a different `.cymem` file need to be created for each configuration. In **Configuration** field, provide an appropriate name for the configuration of `.cymem` file, such as "Hybrid at top", "Hybrid at bottom", "Uniform sectors", "3-byte addressing mode", "4-byte addressing mode", etc. to differentiate each configuration. For example, S25FS256S flash device has multiple configurations as shown in Figure 3

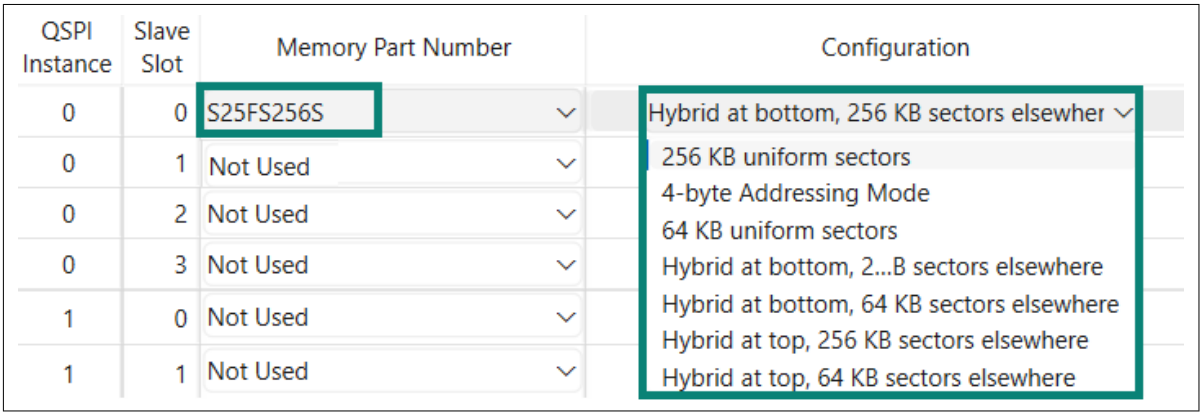

Figure 3 Multiple configurations

***Note**: Slave slot number corresponds to the slave select line that will be connected to the memory device*

4. **Description**: Provide a short description of the memory part for which `.cymem` file is being created

5. **Configuration Is Default**: As mentioned above, a memory part can have multiple `.cymem` files. The file with this check box selected will be the default configuration loaded by the QSPI Configurator when this memory part is selected
6. **General, Commands, Hybrid Erase Sectors tabs**: These sections require data from the memory part datasheet. See the memory part datasheet to fill the remaining data. See the following tables

   a. **General tab**:

Table 1 General → Registers

Field in QSPI Configurator	Datasheet source
`StatusRegisterBusyMask`	Bit position of WIP/Busy. For example, in S25FS128S device datasheet, it is SR1 bit 0 → 0x01
`StatusRegisterQuadEnableMask`	Bit position of Quad enable. For example, in S25FS128S device datasheet, it is SR2 bit1 → 0x02
`StatusRegisterOctalEnableMask`	Bit position of Octal enable (for Octal parts only)
`OctalEnableRegisterAddress`	Octal configuration register address (for Octal parts only)

Table 2 General → Sizes

Field in QSPI Configurator	Datasheet source
`SizeOfMemory`	Total density. For example, 512 Mbit = `0x4000000` bytes. Must cover the whole address range you program
`ProgramPageSize`	Page Program buffer size
`EraseBlockSize`	The sector/block size of the **erase opcode you choose** (e.g., 256 KB uniform, or 4 KB parameter sector), these must match
`NumberOfAddress`	3 for ≤ 16 MB, **4** for parts >16 MB (or when using 4-byte-address opcodes)

Table 3 General → Timings

Field in QSPI Configurator	Datasheet source
`EraseTime`	Sector erase time. For example, in S25FS128S device datasheet, it is tSE
`ChipEraseTime`	Chip/bulk erase time. For example, in S25FS128S device datasheet, it is tCE
`ProgramTime`	Page program time. For example, in S25FS128S device datasheet, it is tPP
`FrequencyOfOperation`	Max. SCK the read command supports (SDR vs DDR limits)

***Note**: For `EraseTime`, `ChipEraseTime`, and `ProgramTime`, use maximum values from the timing table. Not typical value*

Table 4 General → Latency Cycles

Field in QSPI Configurator	Datasheet source
`Mask`	Latency cycles mask
`RegisterAddress`	Latency cycles configuration register address

***Note**: These fields are applicable only if the part uses a variable-latency (dummy-cycle)*

1. In **General → Octal DDR Enable Sequence** is by default set to NA. Update these values if the memory part provides a command sequence to enable Octal DDR mode

b. **Commands tab**: The command table defines each SPI transaction. The command set used differs based on the number of data lines used for SPI

If SPI-MISO:MOSI Data[x:x] is selected as data lines, single SPI commands are used. If Dual SPI-Data[x:x] is selected, Dual SPI commands are used. Similarly, Quad SPI-Data[x:x] and Octal SPI-Data[x:x] use Quad SPI and Octal SPI Commands respectively

For example, S25FS128S device supports up to 4 data lines (Quad SPI). So, Single SPI or Dual SPI or Quad SPI commands can be used depending on the number of data lines used in the application.

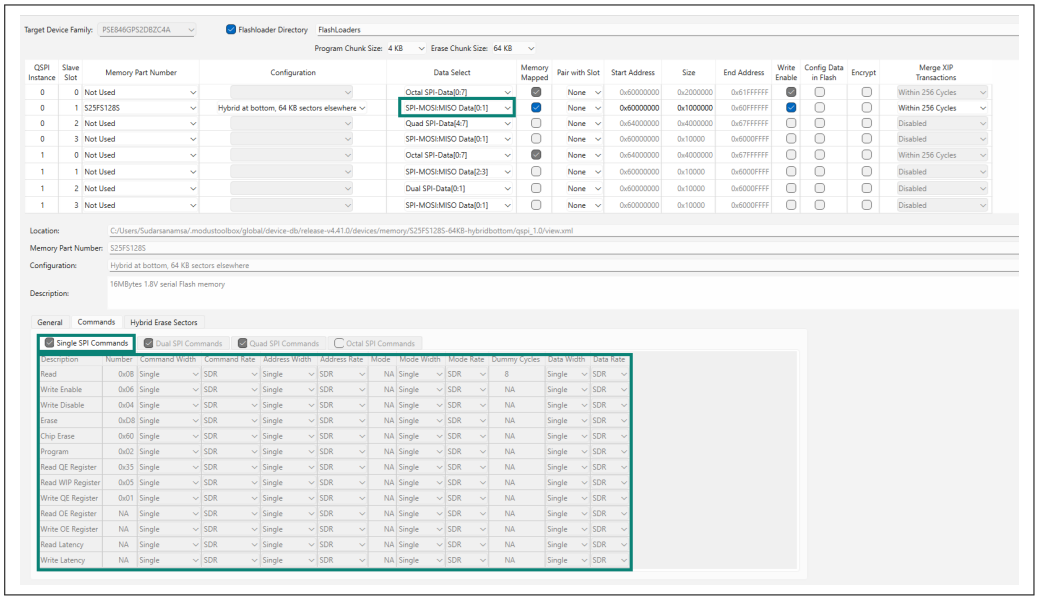

Figure 4 Single SPI commands

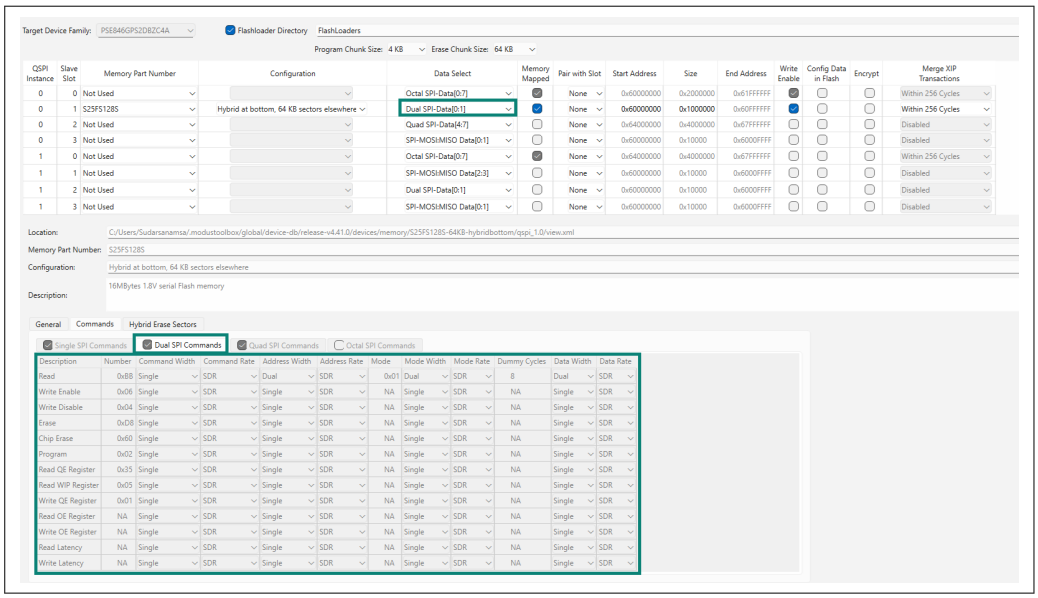

Figure 5 Dual SPI commands

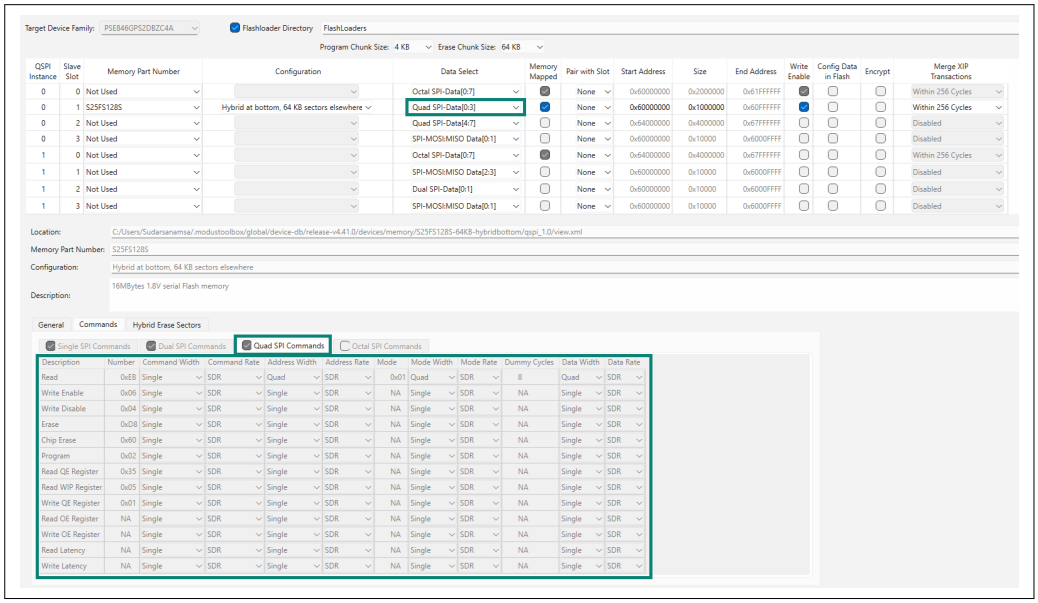

Figure 6 Quad SPI commands

1. Refer to the memory part datasheet to fill the opcodes for different commands in the **Commands** tab. The command names are generic and self-explanatory. However, see the following table for reference

Table 5 Commands to be defined

Command	Description
`Read`	Command to read memory
`Write Enable`	Command to enable write/program/erase
`Write Disable`	Command to disable write
`Erase`	Command for sector erase

(table continues...)

Table 5 (continued) Commands to be defined

Command	Description
`Chip Erase`	Command for bulk erase
`Program`	Command for Page program
`Read QE Register`	Command to read configuration register
`Read WIP register`	Command to read status register
`Write QE Register`	Command to write configuration register
`Read OE Register`	Default value is NA. Applicable only when Octal SPI is available
`Write OE Register`	Default value is NA. Applicable only when Octal SPI is available
`Read Latency`	Default value is NA. Applicable only when Octal SPI is available
`Write Latency`	Default value is NA. Applicable only when Octal SPI is available

***Note**: For some memory parts, the commands code may vary based on the address width. For example, in S25FS128S device, the command for Program Page is 0x02 for both 3-byte or 4-byte addressing mode and 0x12 exclusively for 4-byte addressing mode. Ensure to use all the command codes of the same addressing mode for the configuration to work correctly*

2. For each command, ensure to fill the **`CmdWidth`** / **`AddrWidth`** / **`ModeWidth`** / **`DataWidth`**as defined by the command

c. **Hybrid Erase Sectors tab**: This section is valid only if the memory part supports multiple sectors and if the configuration specifies splitting of memory as per sector architecture mentioned in the device datasheet. If memory is not divided into sectors or uniform architecture is used, this tab can be ignored

If hybrid sectors are used in the configuration, enable them by clicking on the **Hybrid Erase Sectors** check box in this tab. Add a new sector, by clicking on the **Add Row** button

For each sector, the start address, sector count, sector size, erase command of the given sector, erase time of the sector, and region end address need to be mentioned. Note that the sectors need to be contagious and must match the sector architecture mentioned in the memory datasheet

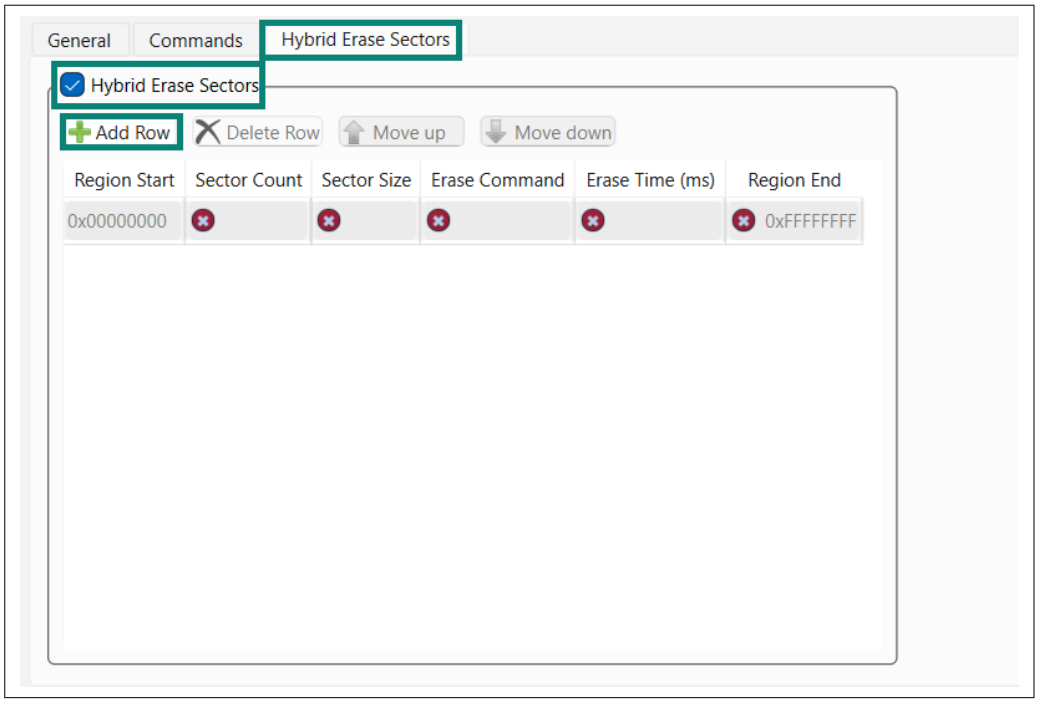

Figure 7 Hybrid Erase Sectors

For example, S25FS128S device memory supports hybrid mode. The memory datasheet provides details of the sector size, count, and sector address range of the different configurations supported by the memory part. This sector data need to be added to the configuration as shown in the following figure

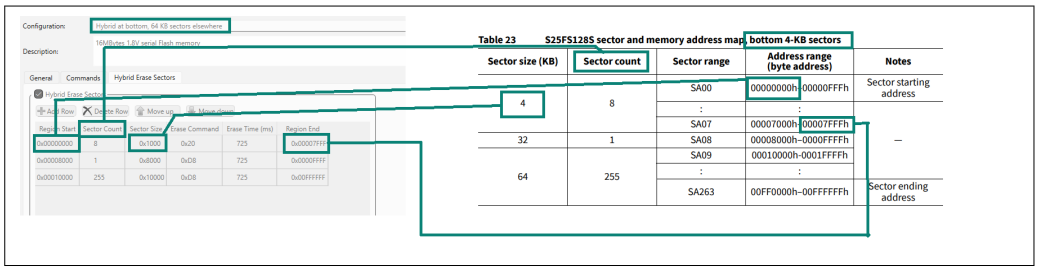

Figure 8 Sector data

***Notes**:*

1. *Sector size for 4 KB = 4 × 1024 = 4096 bytes = 0x1000*
2. *The markup in Figure 8 is shown only for 4 KB sectors. The remaining two sector data for the bottom hybrid configuration also need to be filled similarly for the configuration to work*

If the datasheet provides different erase command for different sectors, add these values as well

7. Once the configuration details are filled, save the `.cymem` file. Close the QSPI Configurator and reopen the QSPI Configurator to use the newly created `.cymem` file
8. If the `.cymem` file needs to be updated, click on **File** > **Open*.cymem File...**

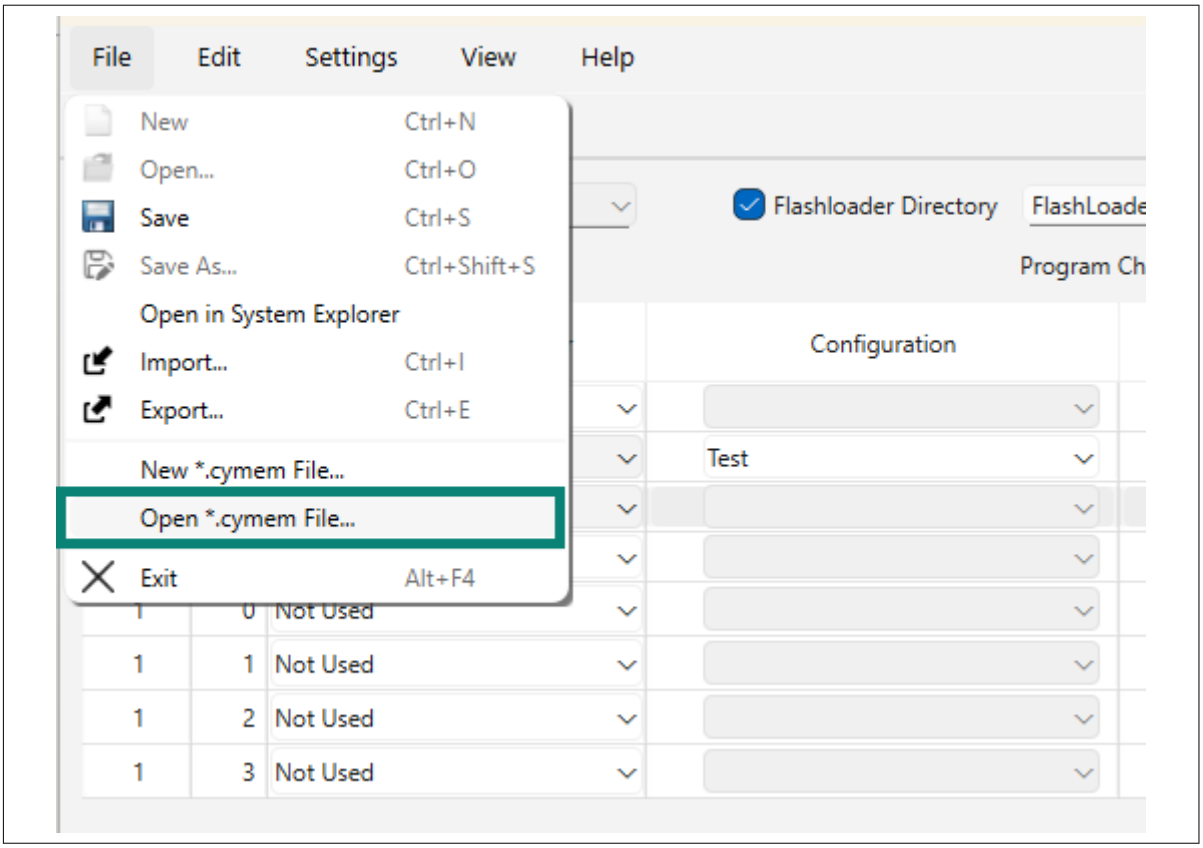

Figure 9 Open an existing .cymem file

9. Browse and select the required `.cymem` file. Once the configuration changes are done, **Save** and close the `.cymem` file. Close the QSPI Configurator and reopen the QSPI Configurator to use the updated `.cymem` file

A new `.cymem` file has been created for the S25FS128S device with uniform sector architecture and making use of only SDR commands. The configurations done for the same is shown in the following images.

***Note**: This is the simplest use case of QSPI flash*

***Note**: Since uniform sector is used, Hybrid erase sectors are not enabled.*

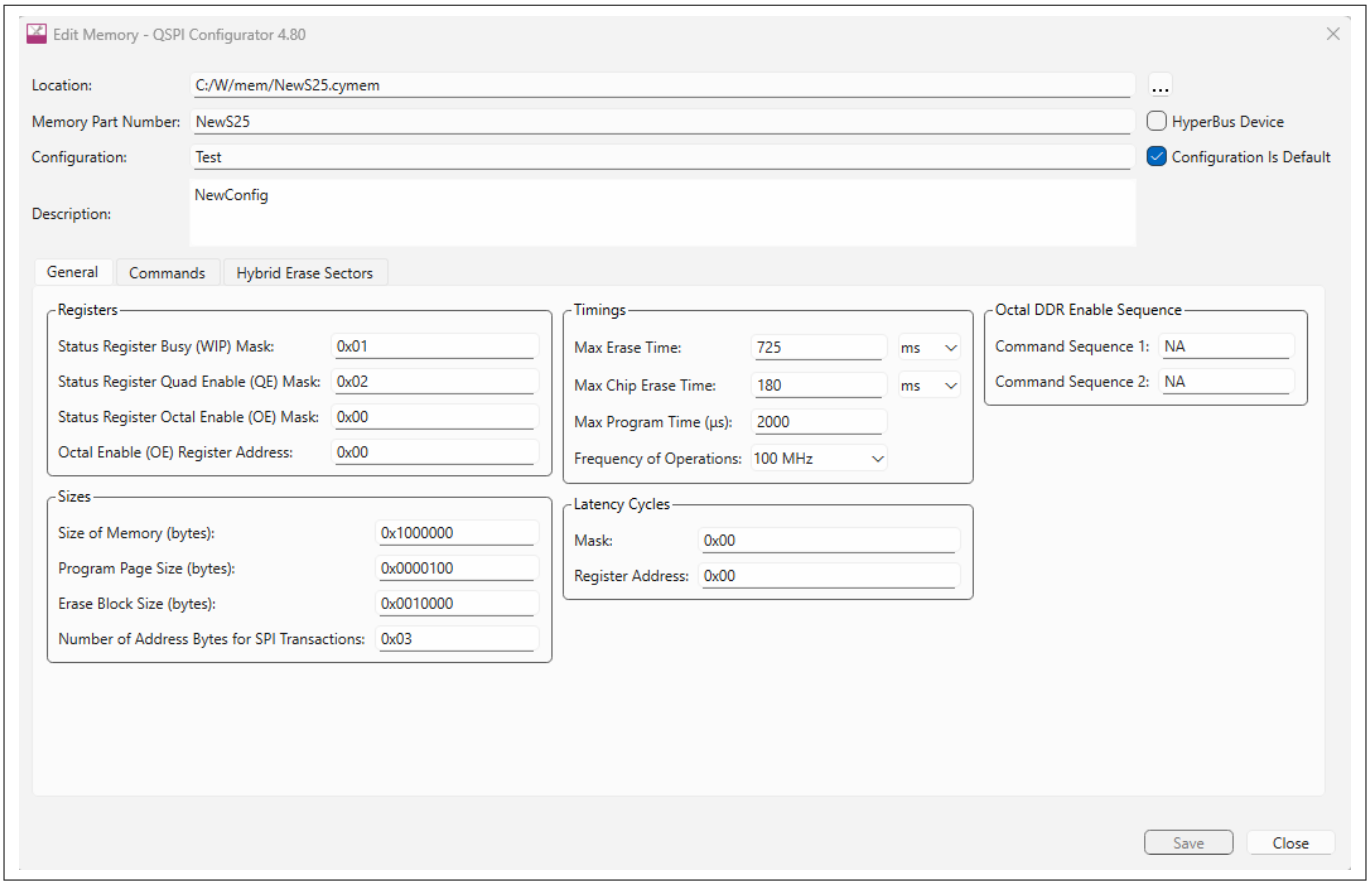

Figure 10 Edit Memory - General tab

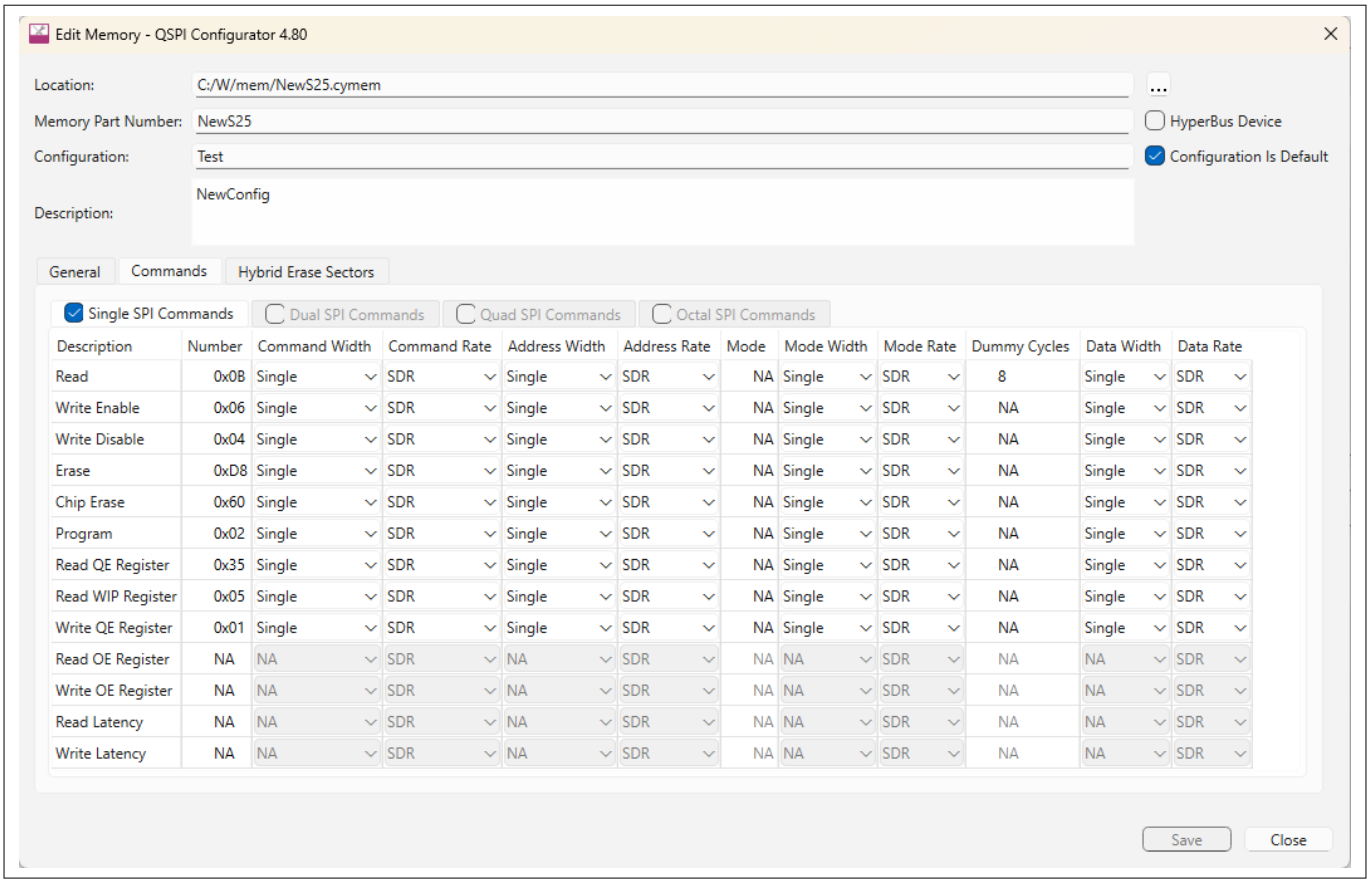

Figure 11 Edit Memory - Commands tab

2.1.2 Adding a HYPERBUS™ device

To add support to a HYPERBUS™ device, such as a HYPERFLASH™ or HYPERRAM™ device, select the **HyperBus Device** as shown in Figure 12. The HYPERBUS™ device has fewer configuration options when compared to non HYPERBUS™ device as HYPERBUS™ uses a fixed 8-bit DDR data bus with a bidirectional read-write data strobe (RWDS).

1. **Location**: Click on the **...** button to specify location to save the CYMEM file
2. **Memory Part Number**: This is the name shown in the drop down in QSPI Configurator
3. **Configuration**: This is invalid for a HYPERBUS™ device as multiple configurations are not possible
4. **Description**: Provide a short description of the memory part for which `.cymem` file is being created
5. **Configuration Is Default**: This is invalid for a HYPERBUS™ device

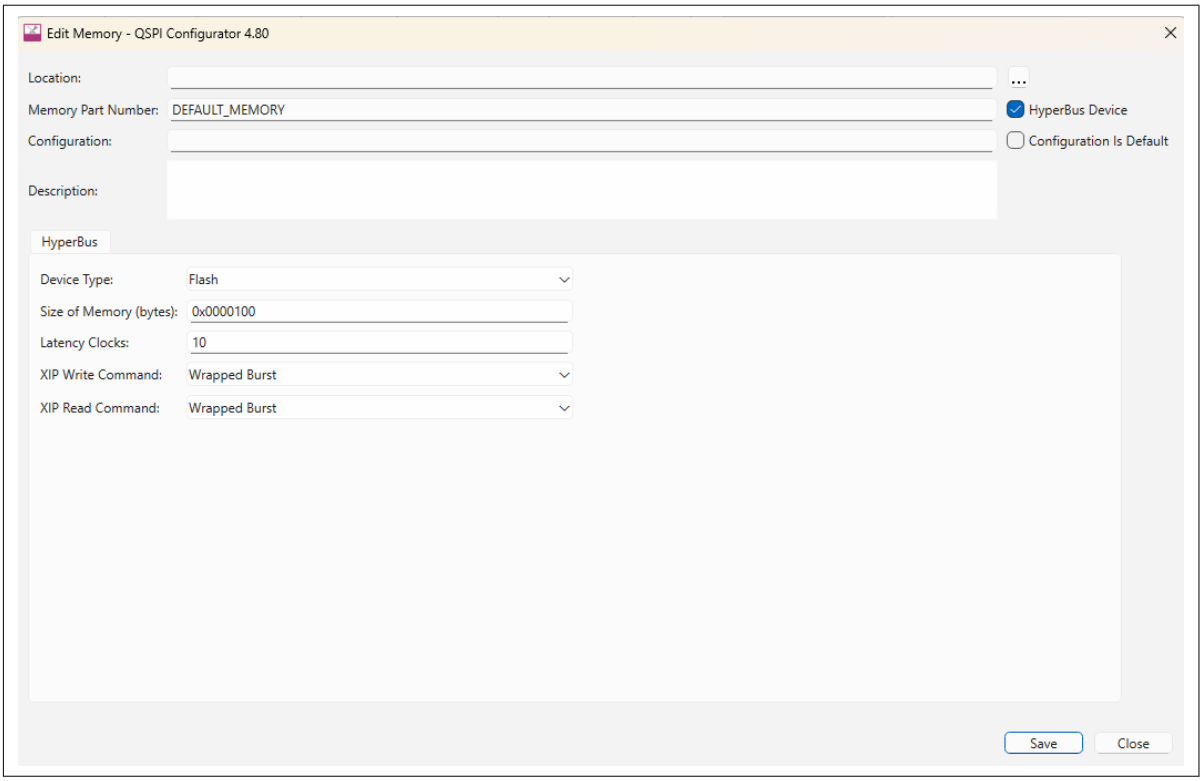

Figure 12 HyperBus Device

6. **HyperBus tab**
   a. **Device Type**: Flash/SRAM depending on the type of HYPERFLASH™ or HYPERRAM™ device being added
   b. **Size of Memory (bytes)**: Device density
   c. **Latency Clocks**: Initial latency count for the operating frequency (Refer device datasheet)
   d. **XIP Write Command**: Wrapped Burst or Continuous Burst
   e. **XIP Read Command**: Wrapped Burst or Continuous Burst
7. Once the configuration details are filled, save the `.cymem` file. Close the QSPI Configurator and reopen the QSPI Configurator to use the newly created `.cymem` file

2.2 Using the CYMEM file

1. Create a new PSOC™ Edge Serial flash read write application for KIT_PSE84_EVAL_EPC2 BSP in a IDE of your choice. For detailed steps, see "Create a new application" section in AN235935 - Getting started with PSOC™ Edge E8 MCU on ModusToolbox™ software
2. Launch the QSPI Configurator
3. Click on **<Browse..>** in the drop down list under **Memory Part Number**

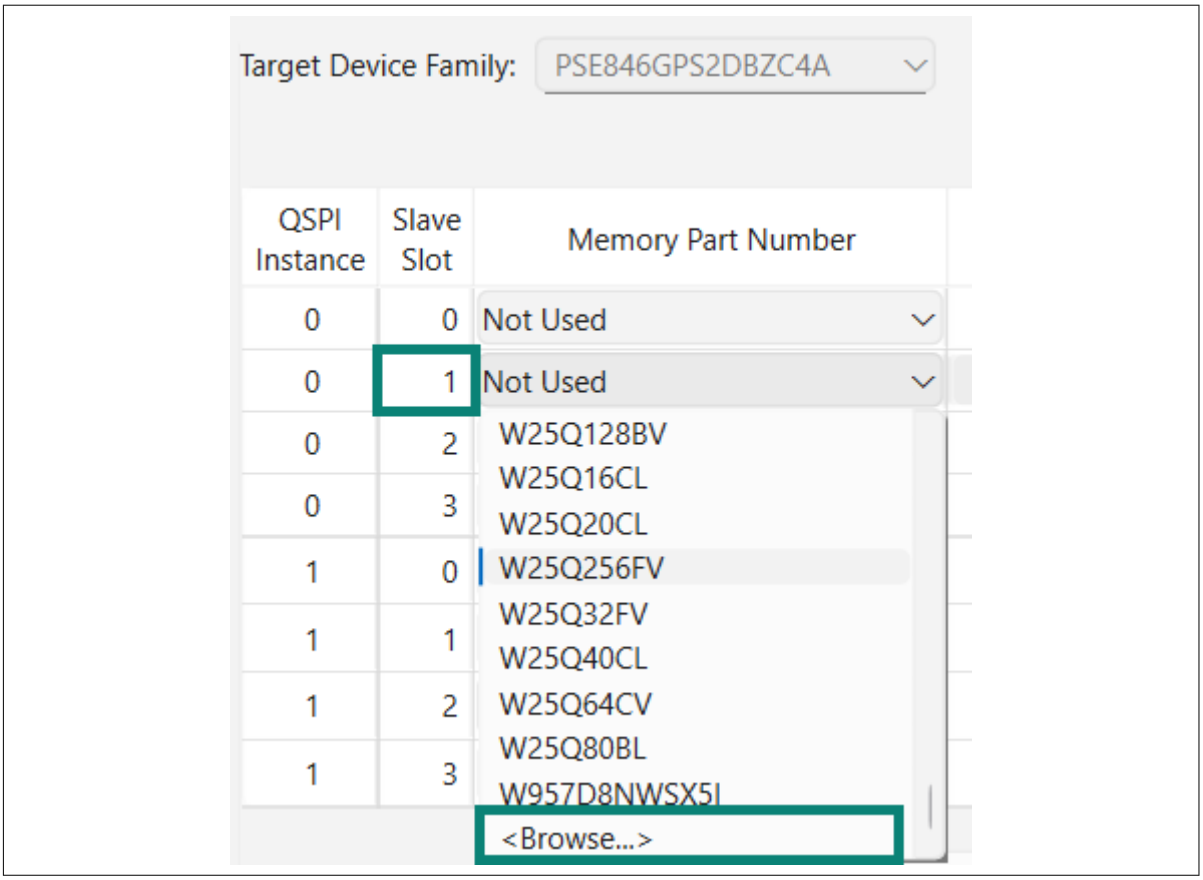

Figure 13 Memory Part Number

***Note**: As mentioned earlier, the slave slot number corresponds to the slave select line that will be connected to the memory device. In KIT_PSE84_EVK_EPC2, it is slave slot 1*

4. Browse and select the new `.cymem` file created
5. The QSPI Configurator can patch flashloaders so that their data structures contain data about the external flash memory that will be connected to your design, and how the memory is configured. To use this feature, select the Flashloader Directory as shown in Figure 14



Figure 14 Flashloader

6. When flashloader is enabled, select appropriate values for Program Chunk Size and Erase Chunk Size
   a. **Program Chunk Size:** Maximum number of bytes sent to the flashloader in one program transfer. Set this to a value supported by the flashloader and compatible with the device's program-page size. Ensure that it does not cross the page boundaries
   b. **Erase Chunk Size:** Maximum region size passed to the flashloader for one erase operation. Set this to an erase granularity supported by the flashloader and the flash device. Ensure that it matches the erase opcode and `Erase Block Size`
7. Configure other settings such as Data select lines, memory mapping, etc. See the ModusToolbox™ QSPI Configurator user guide for more details on the configuration fields
8. Save the configuration. The code gets generated automatically and gets saved in the `GeneratedSource` folder in the BSP folder
9. Build the project
10. Connect the kit to a PC and program the `.hex` file onto the kit

11. Open a serial terminal software, such as Tera Term, and open the COM port connected to the kit
12. On successful erase, write, and read flash operations, the red LED on the kit starts blinking and the status of flash operations is displayed on the terminal window

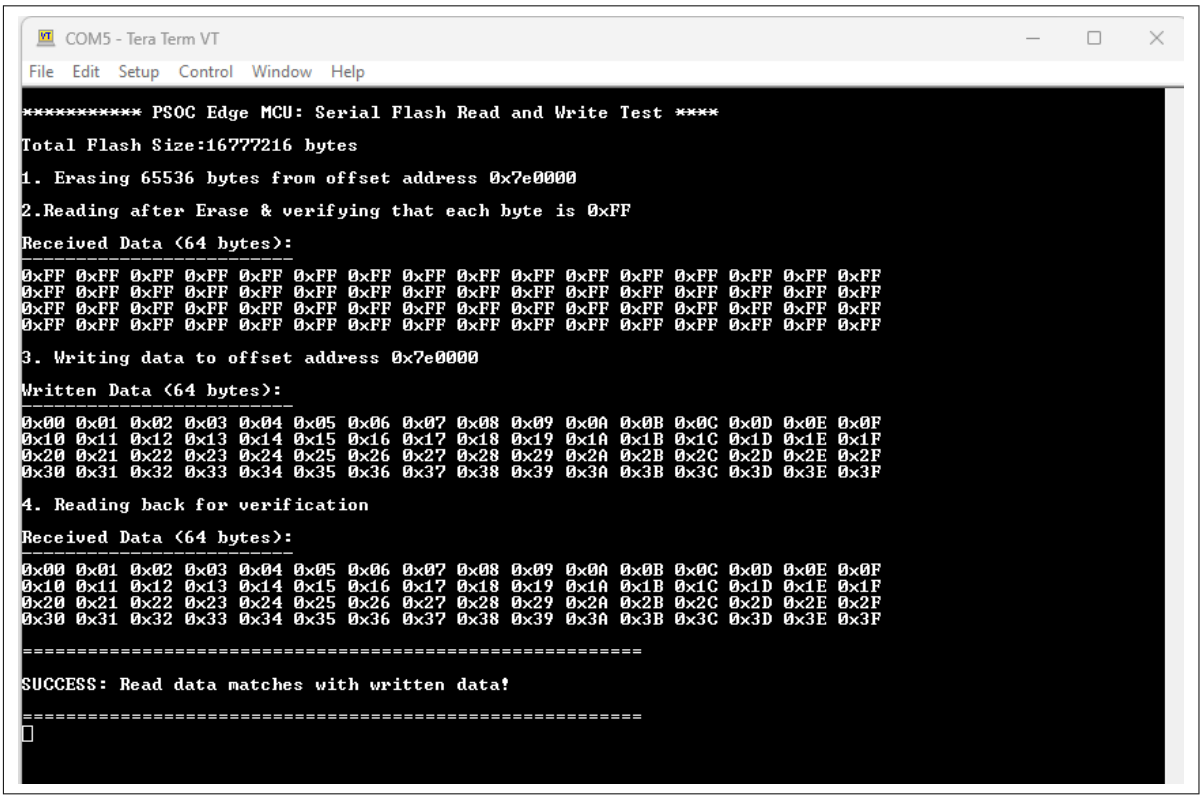

Figure 15 Serial terminal window

This shows that the `.cymem` file is configured correctly and flash operations are functional

2.3 Fine-tuning the CYMEM file

1. Maximizing the read/XIP bandwidth on PSOC™ Edge MCU

   The QSPI NOR flash operates at 106/133/166 MHz on SDR SCLK. PSOC™ Edge MCU has a 400 MHz PLL for the CPU, but the SMIF is operated from the SoCMEM domain, which caps the interface clock to 300 MHz. So, to maximize the data rate, widen the bus (increase data select lines), enable DDR, and turn ON the SMIF cache while operating withing SoCMEM SCLK range

   a. Bandwidth

   Bandwidth = (fSCLK × lanes × rate)/8 MB/s, where

   - lanes = 1 (single), 4 (Quad/ QSPI), 8 (Octal/ OSPI)
   - rate = 1 (SDR) or 2 (DDR/DTR)

Table 6 Clock vs. data rate

Mode at SCLK	Quad (QSPI)	Octal (OSPI)
106 MHz SDR	53 MB/s	106 MB/s
133 MHz SDR	66 MB/s	133 MB/s
166 MHz SDR	83 MB/s	166 MB/s
100 MHz DDR	100 MB/s	200 MB/s

(table continues...)

Table 6 (continued) Clock vs. data rate

Mode at SCLK	Quad (QSPI)	Octal (OSPI)
133 MHz DDR	133 MB/s	266 MB/s

***Note**: The values in Table 6 represent theoretical interface bandwidth and do not account for command overhead, dummy cycles, cache effects, or memory-device limitations*

b. SMIF Clock settings

SMIF0 is operated from the SoCMEM domain. The interface clock is CLK_HF/SMIF_divider. See Table 7 for reference CLK_HF and divider values to achieve target SCLK

Table 7 CLK_HF and divider

Target SCLK	Source CLK_HF	Divider	Best use
100 MHz	200 MHz	2	Quad/Octal DDR (recommended)
133 MHz	266 MHz	2	133 SDR or DDR (if available)
150 MHz	300 MHz	2	166 SDR

2. SMIF configuration best practices for maximum bandwidth

a. **Widest bus the board routes:** OSPI if 8 data lines are available, else QSPI. Avoid single-line reads for XIP

b. **Enable DDR (DTR) read:** Set the read command's `addrRate`/`modeRate`/`dataRate` to DDR

c. **Enable the SMIF cache:** For the XIP region (`Cy_SMIF_InitCache`, write-back/read-allocate). For code execution and random access, this often beats raw clock and it removes repeated opcode+address+dummy latency on hits

d. **Use continuous/burst (XIP) read:** Set the read command's mode byte so subsequent XIP fetches skip the opcode; long sequential bursts amortize command/address/dummy overhead

e. **Trim dummy cycles to the datasheet minimum:** For the chosen SCLK, extra dummy cycles occupy unnecessary bandwidth. Keeping the opcode ↔ dummy ↔ rate consistent is sufficient

f. **Calibrate RX data capture:** At the target SCLK — above ~50 MHz to 80 MHz and for all DDR, enable the SMIF delay-line so that sampling is centered

g. Set 'FrequencyOfOperation` to the bucket matching your SCLK (e.g., `CY_SMIF_100MHZ_OPERATION`)

3. To ensure an error free `.cymem` file, following are the best practices and most common errors while creating a new `.cymem` file

a. If available, start from the vendor `.cymem` for the same family (they usually differ only by density/ sector layout), then adjust density, sector size, timings, and the specific opcodes/dummy cycles per your datasheet and your desired SPI/SDR/DDR mode

b. Dummy cycles are per-opcode. In the S25FS128S device, `0x03` READ = 0 dummy; `0x0B` FAST_READ = 8 dummy (single). Copying an opcode but keeping the wrong latency corrupts every read

c. The size must equal what the opcode actually erases. In the S25FS128S device, `0x20`=4 KB, `0xD8`/`0xDC`=64/256 KB, and the opcode's address width must match `Number of Address Bytes`

d. SizeOfMemory must span the highest address you program (e.g., an app placed at offset 0x100000 needs `SizeOfMemory ≥ 0x100000`

e. Setting Busy Mask as '0' breaks completion polling and hangs during flashing. Ensure this busy or WIP bit position is configured correctly

f. Using typical timeouts instead of max timeouts causes premature hanging. Use max values for all timeouts

2.4 SFDP memories supported in PSOC™ Edge MCU

PSOC™ Edge MCU integrates a limited internal RRAM capacity that is ideal for secure boot, compact bootloaders, and/or early initialization code. PSOC™ Edge MCU can run complex applications. In addition to the internal RRAM, it requires external non-volatile memory for larger code storage and Execute-in-Place (XIP). PSOC™ Edge MCU includes a serial memory interface (SMIF) that can connect to external memories. It supports the following devices:

- SPI family devices:
  - **Bus widths:** single (1-bit), dual (2-bit), Quad (4-bit), Dual-Quad, and Octal (8-bit)
  - **Modes:** SDR (single data rate) and DDR (double data rate)
- HYPERBUS™ devices:
  - HYPERFLASH™ (NOR flash) for high-throughput non-volatile storage
  - HYPERRAM™ (PSRAM/pseudo-SRAM) for high-bandwidth volatile data

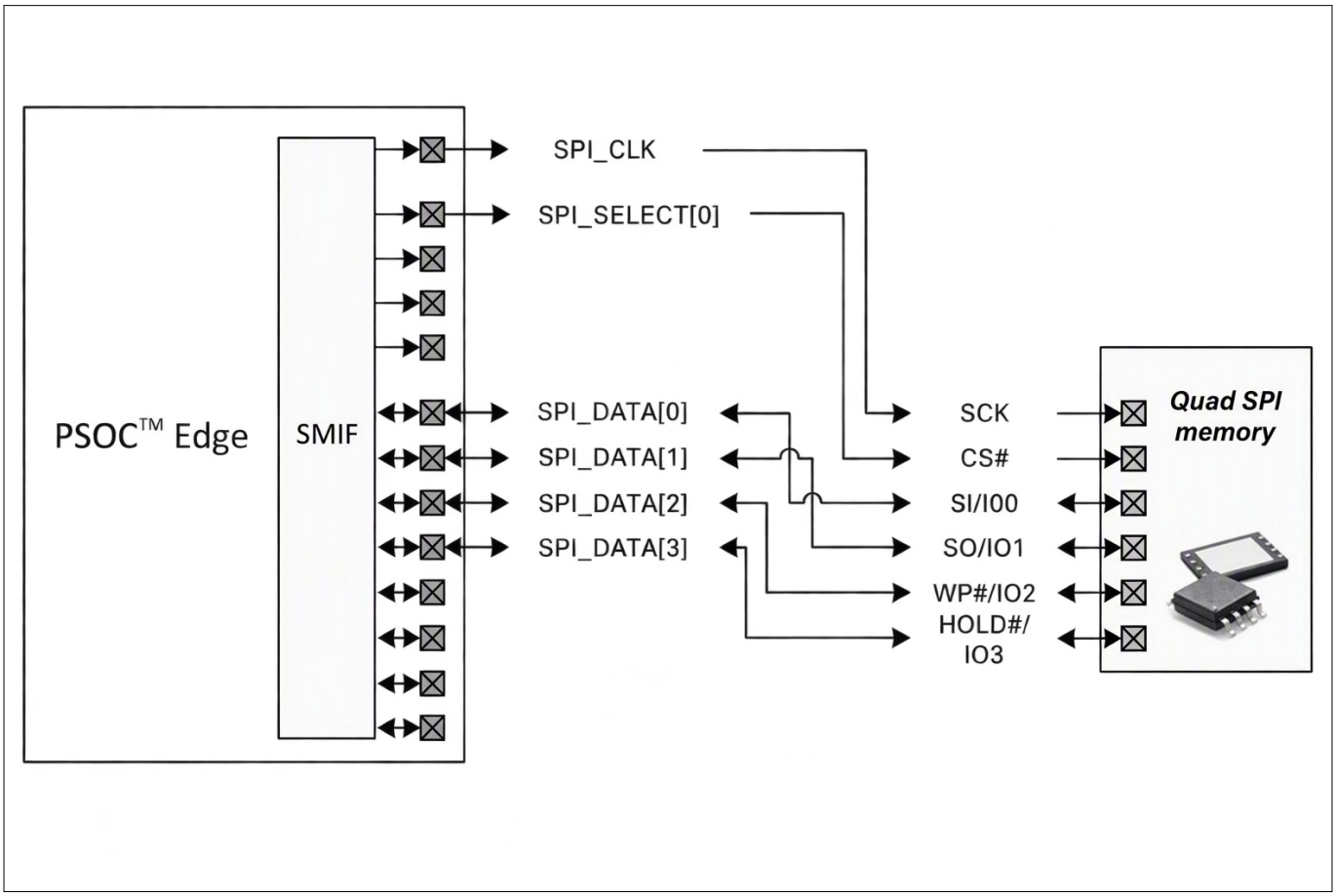

Figure 16 QSPI memory interfaced with SMIF

Extended Boot on PSOC™ Edge MCU supports booting from external memory that implements serial flash discoverable parameters (SFDP) as standardized by JEDEC JESD216B. The SFDP tables describe device commands, bus widths, addressing, dummy cycles, etc., allowing automatic configuration of SMIF in SDR or DDR modes across single/dual/Quad/Octal SPI devices.

Table 8 SFDP compatible external memories and modes are tested and validated on PSOC™ Edge MCUs

Memory device	Bus width	Mode	Speed	Voltage
S25FS128S, S25FS256S [2]	Bus width: single, dual, and Quad	SDR/DDR	Single, dual, and Quad SDR: 133 MHz DDR Quad: 80 MHz	1.7 V to 2.0 V
S25HS512T, S25HS01GT, S25HL512T, S25HL01GT [3]	Bus width: single, dual, and Quad	SDR/DDR	Single, dual, and Quad SDR: 166 MHz DDR Quad: 102 MHz	1.7 V to 2.0 V
S28HS512T, S28HS01GT, S28HL512T, S28HL01GT [4]	Bus width: single and Quad	SDR/DDR	Single SDR: 166 MHz Octal SDR/DDR (HS-T): 200 MHz Octal SDR/DDR (HL-T): 166 MHz	1.7 V to 2.0 V

By default, the extended boot initializes SMIF with QSPI at 50 MHz clock with SDR/DDR set as auto, i.e., use configuration defined by SFDP data. User applications can re-configure SMIF and change the clock frequency. You can re-provision the device and instruct extended boot to boot with Octal SPI. For more details about the re-provisioning, see AN237849 - Getting started with PSOC™ Edge security application note.

For external memories that are not mentioned in the above list or do not implement SFDP (or implement an incomplete/unsupported subset), the recommended approach is to boot from internal RRAM.

Execute a small OEM loader code in internal RRAM to initialize SMIF for the target device, enter the desired bus mode (Quad, Dual-Quad, Octal; SDR/DDR), set latency, and then enable XIP or copy the application image from external memory to internal volatile memory for execution.

3 Implementing DDR using NOR flash

Double Data Rate (DDR), also called Double Transfer Rate (DTR) transfers data on both edges of the SMIF clock, roughly doubling read throughput at the same clock frequency. Standard QSPI/OSPI NOR flash (Winbond, Micron, Infineon, Macronix) all expose dedicated DDR read opcodes in the datasheet.

CYMEM file changes to implement DDR. DDR can be enabled by providing the appropriate read command in the **Commands** tab while generating the `.cymem` file.

1. Enabling DDR is mostly opcode driven. From the memory part datasheet, use the correct DDR/DTR instruction opcode. For example, see Table 9 for the S25FS128S device

Table 9 DDR opcodes in S25FS128S device

Bus mode	DDR opcode
Dual SPI DDR	0xBD
Quad SPI DDR	0xED
Octal SPI DDR	0xEE / 0x9D / 0xFD

Table 10 Configuring Read command in different modes

Field in QSPI Configurator	Single SPI DDR	Quad SPI DDR	Octal SPI DDR
cmdRate	SDR	SDR	DDR
addrRate	SDR	DDR	DDR
modeRate	SDR	DDR	DDR
dataRate	SDR	DDR	DDR
cmdWidth	1	1	8
addrWidth	4	4	8
dataWidth	4	4	8

2. DDR reads use a different (usually higher) dummy-cycle count than the SDR fast read. Use the value from the datasheet's latency table for your target frequency and put it in `DummyCycles` for that opcode. A wrong DDR latency is one of the main reason of shifted/garbled DDR reads
3. Set the **Frequency of Operation** to the bucket matching your SMIF clock
4. DDR mode normally applies only to the read command. Other commands such as erase, program, status commands typically stay in SDR mode. These values can be left unchanged unless the datasheet specifies opcodes for DDR operation

***Note**: The opcode, its phase widths, its phase rates, and its dummy cycles must all match one single row of the datasheet's DDR/DTR read instruction table. Mixing a DDR opcode with SDR rates (or SDR dummy count) leads to corrupt read operation*

Firmware changes to implement DDR

1. In the (secure) init, call `Cy_SysClk_PeriGroupSlaveInit(CY_MMIO_SMIF0_PERI_NR, …_GROUP_NR, …_SLAVE_NR, …_CLK_HF_NR)` so the SMIF core clock (which defines the DDR data-eye) starts running
2. Disable the SMIF before re-configuring
3. Initialize the SMIF config generated from configurator
4. Select the data lines
5. Enable SMIF

6. Set up the serial-memory object. This step sends the device's octal-DDR enable sequence so the memory itself switches into DDR, and programs the DDR read/write opcodes and dummy cycles configured in the `.cymem` file
7. Enable XIP (memory-mapped) mode
8. Enable flash write operation
9. Access memory to read/write at the configured DDR rate

See PSOC™ Edge MCU: PSRAM read, write, and SMIF cache operations code example on how to implement DDR mode.

3.1 Constraints and recommendations of using Octal SPI in DDR mode for PSOC™ Edge MCU

PSOC™ Edge MCU includes a SMIF block that can interface with external SPI-based serial memories across multiple bus widths; single, dual, Quad, dual-Quad, and Octal with SDR and DDR/DTR modes. This flexibility allows you to optimize the bandwidth and pin usage to match the application requirements.

The Octal SPI DDR/DTR memory is suited for high throughput XIP scenarios and large asset streaming because it:

- Provides significantly higher bandwidth for faster XIP and asset streaming
- Improves the effective latency and responsiveness
- Speeds up the device boot time and firmware update time (improves image verification speed)
- Achieves the target throughput at lower I/O clock rates

Operational constraints

Octal DDR/DTR memory configuration operates with a two-byte (16-bit) word granularity (as per JESD251 xSPI standard), while the other configurations use one-byte (8-bit) word granularity. This leads to byte-order swapping within the word if the bus width used during programming differs from the bus width used during execution or reading. For example, if you write (0x01, 0x30, 0xc0, 0xb2, 0x70, 0x47) in Octal DDR/DTR mode and read it back in single SPI mode, the received data will be (0x30, 0x1, 0xb2, 0xc0, 0x47, 0x70).

Switching the SMIF modes post-programming can cause the read operation to return byte-swapped data, resulting in corrupted instructions or misinterpreted assets.

Recommendations

- If you program the external memory while the SMIF is configured for Octal DDR/DTR, you must read/execute with the SMIF in Octal DDR/DTR
- If you program the external memory while the SMIF is configured for single/dual/Quad/Octal SDR, you must also read/execute in the same mode you used for programming

The recommendation is to not change the SMIF bus-width/transfer configuration after programming the content. Choose the required mode/bus width and keep the configuration consistent for programming and runtime access of external memory.

4 QSPI vs. OSPI performance comparison

This section compares the serial-memory benchmark results compared using the PSOC™ Edge MCU applications. The comparison includes erase, write, read, and interrupt callback timings. The KIT_PSE84_EVK_EPC2 has a QSPI part, S25FS128S has an OPSI part, and S70KS1283 device connected to the PSOC™ Edge MCU.

The performance benchmark uses the Cortex®-M DWT cycle counter and converts cycles using the reported 200 MHz core clock. Both QSPI and OSPI were compared against the time takes for a 4 KB read/write operation.

The interrupt benchmark compares the SysTick interrupt callback execution time when executed from internal RAM when compared to XIP. For time measurement, a 64 callback samples with a 64-iteration workload was used.

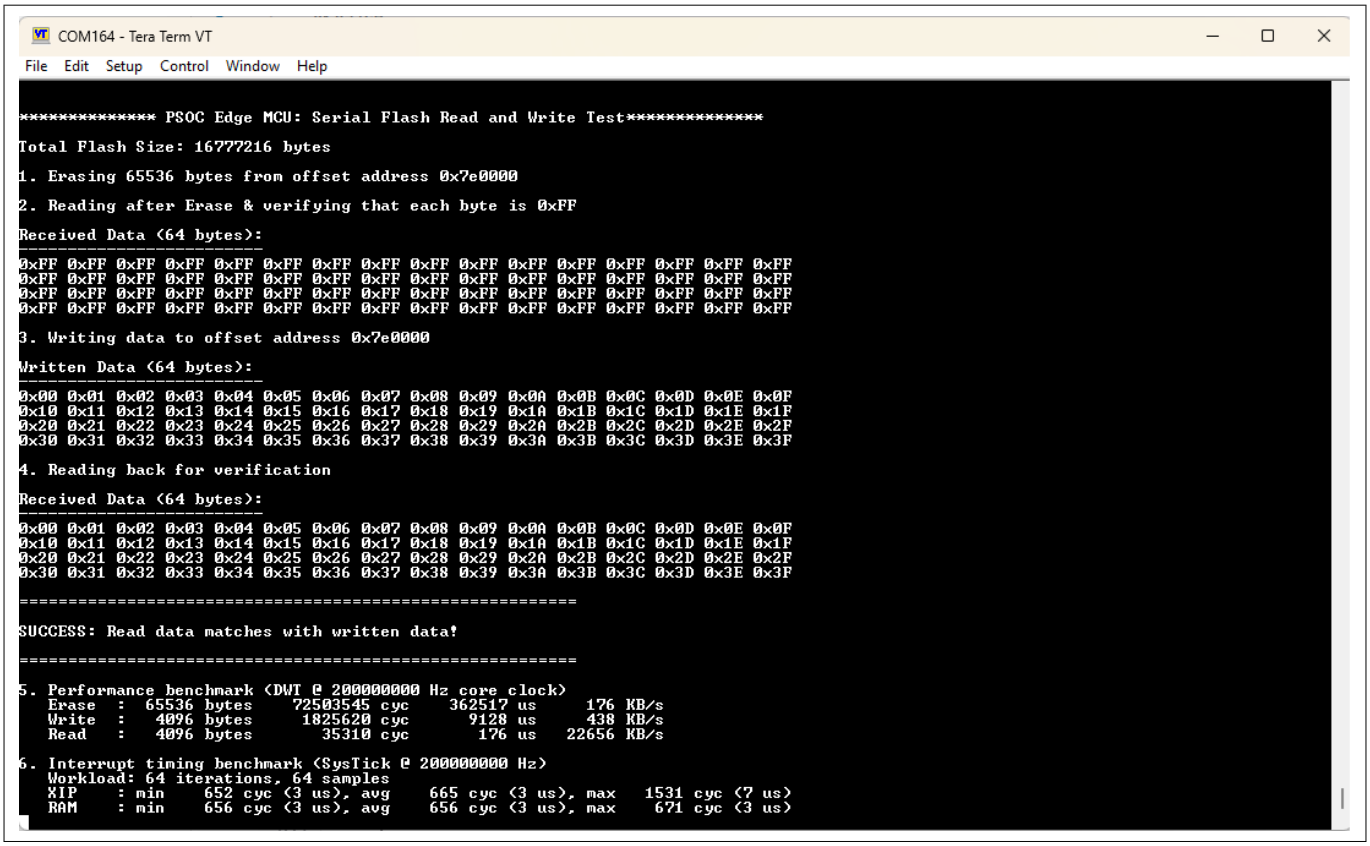

Figure 17 QSPI results

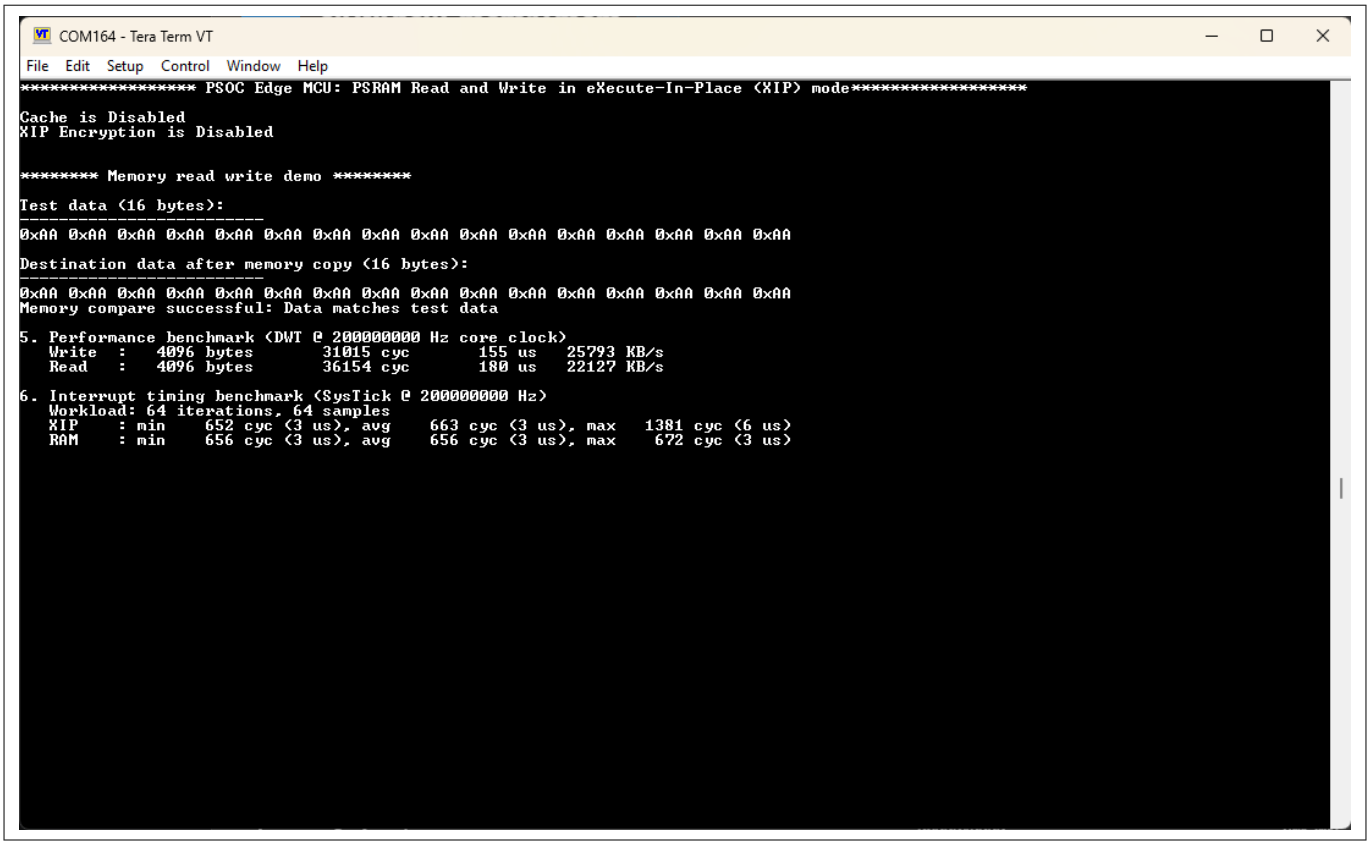

Figure 18 OSPI results

1. **Read performance:** The QSPI read measured 176 µs for 4 KB, compared with 180 µs for the OSPI/XIP run. The measured times are very close, with QSPI approximately 4 µs faster in this capture. For short or moderate transfers like a 4 KB block, non-payload phases, such as sending the instruction command, address bits, and waiting through mandatory dummy cycles or flash internal access latency; consume a fixed amount of time. As the selected QSPI and OSPI part have similar dummy cycle constraints, the percentage of time spent on overhead can make the net read times feel similar despite the OPSI's speed advantage. Also, in OSPI, read was performed in MMIO mode. The throughput of OSPI in XIP mode is 4,744 KB/s. The throughput is affected by many parameters and must be taken care appropriately for required flash performance

   Read throughput in MMIO mode:

   a. QSPI: 22,658 KB/s
   b. OSPI: 22,127 KB/s

   KIT_PSE84_EVK_EPC2 supports QSPI with up to 133 MHz operating speed in SDR mode allowing a throughput of 66.5 MB/s whereas OSPI can support up to 200 MHz DDR allowing up to 400 MB/s throughput
2. **Write performance:** The OSPI run was substantially faster for the 4 KB write (approximately 58.9 times faster)
   a. QSPI: 9,128 µs, 438 KB/s
   b. OSPI: 155 µs, 25,793 KB/s
3. **Interrupt timing:** The interrupt timing results are nearly identical for the two runs. Internal RAM is stable at approximately 656 cycles average, while XIP has a similar average of 663 to 665 cycles but a higher maximum caused by occasional external-memory fetch latency.

The maximum callback time was:

a. QSPI: 1,531 cycles (7 µs)
b. OSPI: 1,381 cycles (6 µs)

The interrupt benchmark measures callback execution after the SysTick handler has dispatched the callback. It does not include the complete interrupt entry and return overhead. The RAM result is therefore the baseline for the callback workload when code is fetched from internal memory

4.1 Use cases and selection guidance

QSPI and OSPI are interface protocols. The flash device technology, command set, clock rate, data width, and memory controller configuration all affect the final performance. The choice should therefore be based on the required capacity, bandwidth, boot support, cost, and software complexity.

Table 11 Memory selection

Requirement	Recommended memory
Lowest-complexity persistent boot or data storage	QSPI NOR flash
Persistent storage with higher XIP/read bandwidth	OSPI NOR flash
Fast large volatile working memory	OSPI PSRAM
Frequent writes that must survive power loss	NOR flash, with wear management
Frequent high-volume temporary writes	OSPI PSRAM
Sector erase and page programming	QSPI or OSPI NOR flash

4.1.1 QSPI NOR flash

QSPI NOR flash is generally the better choice when non-volatile storage and broad compatibility are the main requirements.

Use QSPI NOR flash for:

1. Boot images and firmware storage where the board already has QSPI support
2. Configuration data, calibration data, assets, and logging that must survive power loss
3. Moderate-size code or data storage where read throughput is sufficient
4. Designs that prioritize lower pin count, lower device cost, and a mature software ecosystem
5. Applications that require sector erase and page-program operations

The measured QSPI run demonstrates the expected flash behavior: reads are fast, but programming and especially erase are much slower because the flash array must be programmed or erased internally. QSPI is a good choice for persistent storage when the application does not require the highest XIP bandwidth.

4.1.2 OSPI NOR flash

OSPI NOR flash is preferable when the application needs non-volatile storage but also needs higher read bandwidth or faster XIP instruction fetches than a conventional QSPI design can provide.

Use OSPI NOR flash for:

1. Large firmware images and high-capacity boot storage
2. XIP applications with substantial code-fetch or read-bandwidth requirements
3. Graphics, machine-learning models, media assets, or lookup tables that are read frequently
4. Systems where a wider data bus can reduce read time without moving the data into internal RAM
5. Products designed around an OSPI-capable memory controller and an OSPI-qualified boot flow

OSPI NOR flash still has flash-specific program and erase delays. A wider bus can improve transfer time, but it does not remove the device's page-program or sector-erase latency.

4.1.3 OSPI PSRAM

OSPI PSRAM is the better choice when high-throughput volatile working memory is needed rather than persistent storage.

Use OSPI PSRAM for:

1. Large runtime buffers that do not fit in internal RAM
2. Frame buffers, audio buffers, packet buffers, and temporary processing data
3. Memory-mapped data structures requiring frequent reads and writes
4. XIP-style workloads where the memory is used for fast, directly addressed execution or data access and persistence is not required

PSRAM writes are memory transactions, while NOR flash writes require a program operation and status polling. PSRAM does not provide flash persistence and normally has no sector-erase operation.

5 Exclusions

The QSPI Configurator supports serial NOR flash and compatible serial RAM devices. The following memory types are not supported, because they require different protocols, controllers, or management logic:

1. **NAND flash:** Requires bad-block management, ECC, and page-based access that the QSPI NOR model does not provide
2. **eMMC and SD cards:** Use SD/MMC command protocols and dedicated controllers, not QSPI memory-mapped access
3. **Parallel NOR/NAND hardware limitations:** Requires multiple address/data and control pins, which are incompatible with the serial QSPI bus
4. **DRAM (SDRAM, DDR, LPDDR):** Require refresh cycles, calibration, and a dedicated memory controller/PHY
5. **Serial devices with unsupported protocols:** Memories using nonstandard opcodes, addressing, dummy-cycle behavior, or signaling modes not represented in the configurator's device model

To conclude, only memories that follow a supported serial QSPI protocol and command set can be configured; devices needing a separate controller or special management are not configurable through QSPI Configurator.

References

[1] Infineon Technologies AG: *PSOC™ Edge MCU*webpage; Available online
[2] Infineon Technologies AG: *S25FS128S, S25FS256S, 128 Mb (16 MB) / 256 Mb (32 MB) FS-S Flash SPI Multi-I/O, 1.8 V*; Available online
[3] Infineon Technologies AG: *S25HS512T, S25HS01GT, S25HL512T, S25HL01GT, 512 Mb/1 Gb SEMPER™ flash Quad SPI, 1.8 V/3.0 V*; Available online
[4] Infineon Technologies AG: *S28HS512T, S28HS01GT, S28HL512T, S28HL01GT, S512Mb/1Gb SEMPER™ Flash Octal interface, 1.8 V/3.0 V*; Available online
[5] Infineon Technologies AG: *ModusToolbox™ QSPI Configurator guide*; Available online
[6] Infineon Technologies AG: *PSOC™ E8xxGP device support library*; Available online
[7] Infineon Technologies AG: *PSOC™ Edge MCU: Serial flash read and write*; Available online
[8] Infineon Technologies AG: *PSOC™ Edge MCU: PSRAM read, write, and SMIF cache operations*; Available online

Revision history

Document revision	Date	Description of changes
**	2026-10-09	Initial release

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2026-10-09

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2026 Infineon Technologies AG

All Rights Reserved.

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference

IFX-kzu1788259343823

Important notice

Products which may also include samples and may be comprised of hardware or software or both ("Product(s)") are sold or provided and delivered by Infineon Technologies AG and its affiliates ("Infineon") subject to the terms and conditions of the frame supply contract or other written agreement(s) executed by a customer and Infineon or, in the absence of the foregoing, the applicable Sales Conditions of Infineon. General terms and conditions of a customer or deviations from applicable Sales Conditions of Infineon shall only be binding for Infineon if and to the extent Infineon has given its express written consent.

For the avoidance of doubt, Infineon disclaims all warranties of non-infringement of third-party rights and implied warranties such as warranties of fitness for a specific use/purpose or merchantability. Infineon shall not be responsible for any information with respect to samples, the application or customer's specific use of any Product or for any examples or typical values given in this document.

If this document is marked as 'Preliminary', 'Target', 'Draft' or 'Proposal', Infineon reserves the right to change all information given in this document at any time without notice. Subject to the development and release of a Product for series supply by Infineon, the technical specifications of the Product are set forth in the relevant datasheet provided by Infineon without such marking.

The data contained in this document is exclusively intended for technically qualified and skilled customer representatives. It is the responsibility of the customer to evaluate the suitability of the Product for the intended application and the customer's specific use and to verify all relevant technical data contained in this document in the intended application and the customer's specific use. The customer is responsible for properly designing, programming, and testing the functionality and safety of the intended application, as well as complying with any legal requirements related to its use.

Unless otherwise explicitly approved by Infineon, Products may not be used in any application where a failure of the Products or any consequences of the use thereof can reasonably be expected to result in personal injury.

However, the foregoing shall not prevent the customer from using any Product in such fields of use that Infineon has explicitly designed and sold it for, provided that the overall responsibility for the application lies with the customer. Infineon expressly reserves the right to use its content for commercial text and data mining (TDM) according to applicable laws, e.g. Section 44b of the German Copyright Act (UrhG). If the Product includes security features:

Because no computing device can be absolutely secure, and despite security measures implemented in the Product, Infineon does not guarantee that the Product will be free from intrusion, data theft or loss, or other breaches ("Security Breaches"), and Infineon shall have no liability arising out of any Security Breaches.

If this document includes or references software:

The software is owned by Infineon under the intellectual property laws and treaties of the United States, Germany, and other countries worldwide. All rights reserved. Therefore, you may use the software only as provided in the software license agreement accompanying the software.

If no software license agreement applies, Infineon hereby grants you a personal, non-exclusive, non-transferable license (without the right to sublicense) under its intellectual property rights in the software (a) for software provided in source code form, to modify and reproduce the software solely for use with Infineon hardware products, only internally within your organization, and (b) to distribute the software in binary code form externally to end users, solely for use on Infineon hardware products. Any other use, reproduction, modification, translation, or compilation of the software is prohibited.

For further information on the Product, technology, delivery terms and conditions, and prices, please contact your nearest Infineon office or visit www.infineon.com.