

ModusToolbox™ QSPI Configurator user guide

ModusToolbox™ tools package version 3.1.0

QSPI Configurator version 4.10

About this document

Scope and purpose

The Quad Serial Peripheral Interface (QSPI) Configurator is a stand-alone tool included with the ModusToolbox™ software intended to open or create configuration files, configure memory slots, and generate code for your application.

Intended audience

This document helps application developers understand how to use the QSPI Configurator as part of creating a ModusToolbox™ application.

Document conventions

Convention	Explanation
Bold	Emphasizes heading levels, column headings, menus and sub-menus
<i>Italics</i>	Denotes file names and paths.
Courier New	Denotes APIs, functions, interrupt handlers, events, data types, error handlers, file/folder names, directories, command line inputs, code snippets
File > New	Indicates that a cascading sub-menu opens when you select a menu item

Abbreviations and definitions

The following define the abbreviations and terms used in this document:

- MPN – memory part number
- QSPI – quad serial peripheral interface
- SFDP – serial flash discoverable parameter
- SMIF – serial memory interface

Reference documents

Refer to the following documents for more information as needed:

- [Device Configurator user guide](#)
- [PDL reference guide](#)
- Device datasheets
- Device technical reference manuals

Table of contents

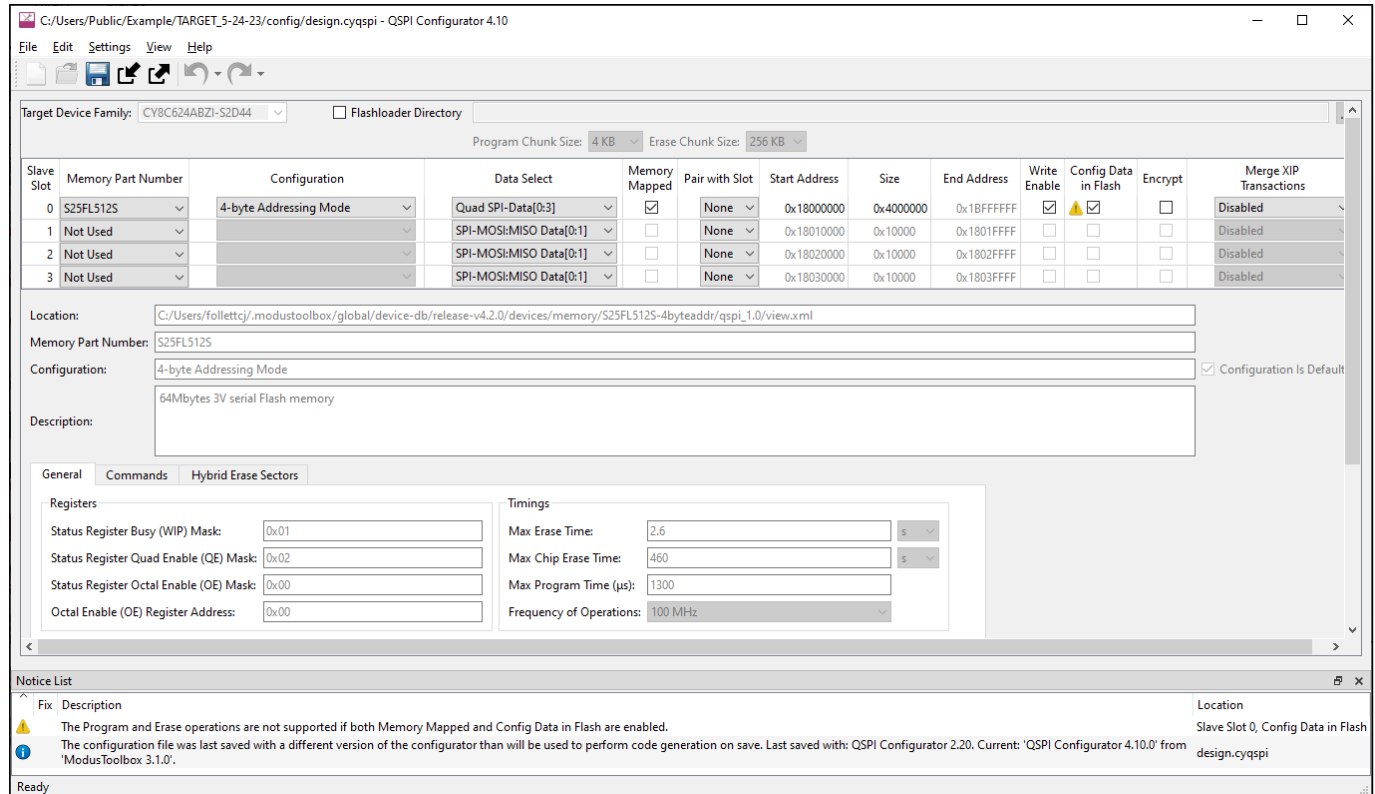
Table of contents

- 1 Overview..... 3**
- 1.1 Supported software3
- 2 Launch the QSPI Configurator..... 4**
- 2.1 make command4
- 2.2 VS Code and Eclipse4
- 2.3 From the Device Configurator.....4
- 2.4 Executable (GUI).....5
- 2.5 Executable (CLI).....5
- 3 Quick start..... 6**
- 4 Code generation..... 7**
- 4.1 Files used by the configuration tool7
- 5 GUI description 8**
- 5.1 Menus.....8
- 5.2 Toolbar9
- 5.3 Target Device Family9
- 5.4 Flashloader Directory9
- 5.5 QSPI configuration fields10
- 5.6 Edit memory file fields11
- 6 Create new memory file14**
- 6.1 Memory database.....14
- 7 XML schemas15**
- 7.1 *.cyqspi schema15
- 7.2 *.cymem schema.....15
- 8 Version changes17**

Overview

1 Overview

The QSPI Configurator is intended to create and configure QSPI configuration for external memory chips and generate code to control the application firmware.



Some devices allow you to extend the default RAM and flash with external memory chips of the following types: flash, non-volatile FRAM, or RAM. Use the SPI to connect a chip to the device. You can attach up to four external memory parts and configure them with the QSPI Configurator.

The QSPI Configurator bears information about the set of default memory parts. However, if your specific memory part is not present, you can create a new one. In the QSPI Configurator, you can select an attached memory part or Serial Flash Discoverable Parameter (SFDP), which is a standard to query the information about attached memory parts at run-time, for each slot and configure the slot parameters.

1.1 Supported software

Name	Version	Link
Peripheral Driver Library	3.3.1	GitHub repo: https://github.com/Infineon/mtb-pdl-cat1 SMIF Driver Documentation: https://Infineon.github.io/mtb-pdl-cat1/pdl_api_reference_manual/html/group_group_smif.html

Launch the QSPI Configurator

2 Launch the QSPI Configurator

There are numerous ways to launch the QSPI Configurator, and those ways depend on how you use the various tools in ModusToolbox™ software.

2.1 make command

As described in the [ModusToolbox™ tools package user guide](#) build system chapter, you can run numerous make commands in the application directory, such as launching the QSPI Configurator. After you have created a ModusToolbox™ application, navigate to the application directory and type the following command in the appropriate bash terminal window:

```
make qspi-configurator
```

This command opens the QSPI Configurator GUI for the specific application in which you are working.

2.2 VS Code and Eclipse

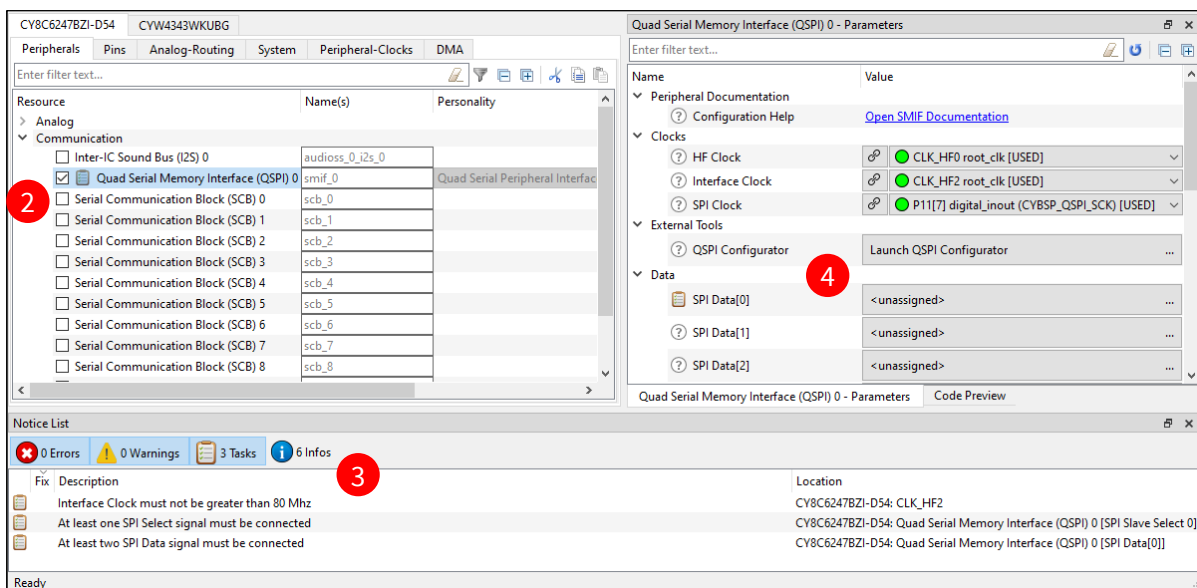
VS Code and Eclipse have tools to launch the BSP Assistant from within an open application. Refer to the applicable user guide for more details:

- [VS Code for ModusToolbox™ user guide](#)
- [Eclipse IDE for ModusToolbox™ user guide](#)

2.3 From the Device Configurator

You can also launch the QSPI Configurator from the Device Configurator:

1. Open the Device Configurator. See the [Device Configurator user guide](#) for details.
2. On the **Peripherals** tab, enable the **QSPI** resource.
3. Optionally, review and resolve the tasks in the Notice List pane. These can be resolved later.
4. On the **Parameters** tab, click the **Launch QSPI Configurator** button.



Launch the QSPI Configurator

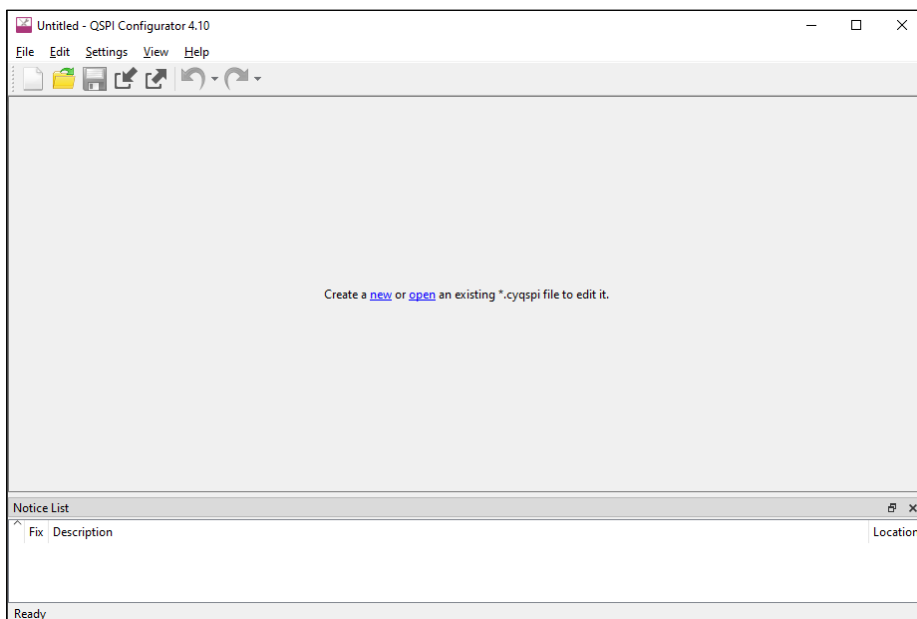
The QSPI Configurator saves configuration information in a *.cyqspi file, which contains all the required information about the device and the application. When you save changes, the QSPI Configurator generates/updates firmware in the BSP's "GeneratedSource" folder.

2.4 Executable (GUI)

If you don't have an application or if you just want to see what the configurator looks like, you can launch the QSPI Configurator GUI by running its executable as appropriate for your operating system (for example, double-click it or select it using the Windows **Start** menu). By default, it is installed here:

```
<Install_dir>/ModusToolbox/tools_<version>/qspi-configurator-<version>
```

When launched this way, the QSPI Configurator opens without any memories configured.



You can either open a specific Configuration file or create a new one. See [Menus](#) for more information.

2.5 Executable (CLI)

The QSPI Configurator executable can be run from the command line, and it also has a "cli" version of the executable as well. Running configurator executables from the command line can be useful as part of batch files or shell scripts to re-generate the source code based on the latest configuration settings. The exit code for the executable is zero if the operation is successful, or non-zero if the operation encounters an error. For more information about the command-line options, run the executable using the -h option.

Quick start

3 Quick start

This section provides a simple workflow for how to use the QSPI Configurator.

1. [Launch the QSPI Configurator.](#)
2. Open an existing or create a new configuration file.
3. Specify the configuration file location.

Note: The device database (device-db) is loaded before the configuration. If device-db is not present on the disk, it will be downloaded automatically. Observe the notification on the status bar.

4. Select a memory from the **Memory Part Number** list.
 - a. If your memory device supports SFDP, select **Auto detect SFDP**.

Slave Slot	Memory Part Number	Configuration	Data Select
0	Auto detect SFDP	<Not Applicable>	Quad SPI-Data[0:3]
1	Not Used	<Not Applicable>	SPI-MOSI:MISO Da
2	Auto detect SFDP	<Not Applicable>	SPI-MOSI:MISO Da
3	23A1024	<Not Applicable>	SPI-MOSI:MISO Da
	23LCV1024	<Not Applicable>	SPI-MOSI:MISO Da
	CY15B102Qx	<Not Applicable>	SPI-MOSI:MISO Da

- b. If your memory device does not support SFDP, or for manual configuration, select a memory from the **Memory Part Number** list, and specify the configuration parameters as required (for example, memory mapped, write enable, etc.). See [QSPI Configuration Fields](#).

Slave Slot	Memory Part Number	Configuration	Data Select
0	Auto detect SFDP	<Not Applicable>	Quad SPI-Data[0:3]
1	Not Used	<Not Applicable>	SPI-MOSI:MISO Da
2	Auto detect SFDP	<Not Applicable>	SPI-MOSI:MISO Da
3	23A1024	<Not Applicable>	SPI-MOSI:MISO Da
	23LCV1024	<Not Applicable>	SPI-MOSI:MISO Da
	CY15B102Qx	<Not Applicable>	SPI-MOSI:MISO Da
	CY15B104QSN	<Not Applicable>	
	CY15B104Q	<Not Applicable>	

Note: If the required memory part number is not present in the list, you can add it. Select **<Browse>...**, navigate to the memory file location, and select it.

5. Save the Configuration file to generate source code. See [Code generation](#).
6. Use the generated structures as input parameters for QSPI functions in your application.

Code generation

4 Code generation

The QSPI Configurator generates code into the GeneratedSource directory in your BSP, or in the same location you saved the Configuration file for non-Eclipse IDE applications. That directory contains the necessary source (.c) and header (.h) files for the generated firmware, which uses the relevant driver APIs to configure the hardware.

4.1 Files used by the configuration tool

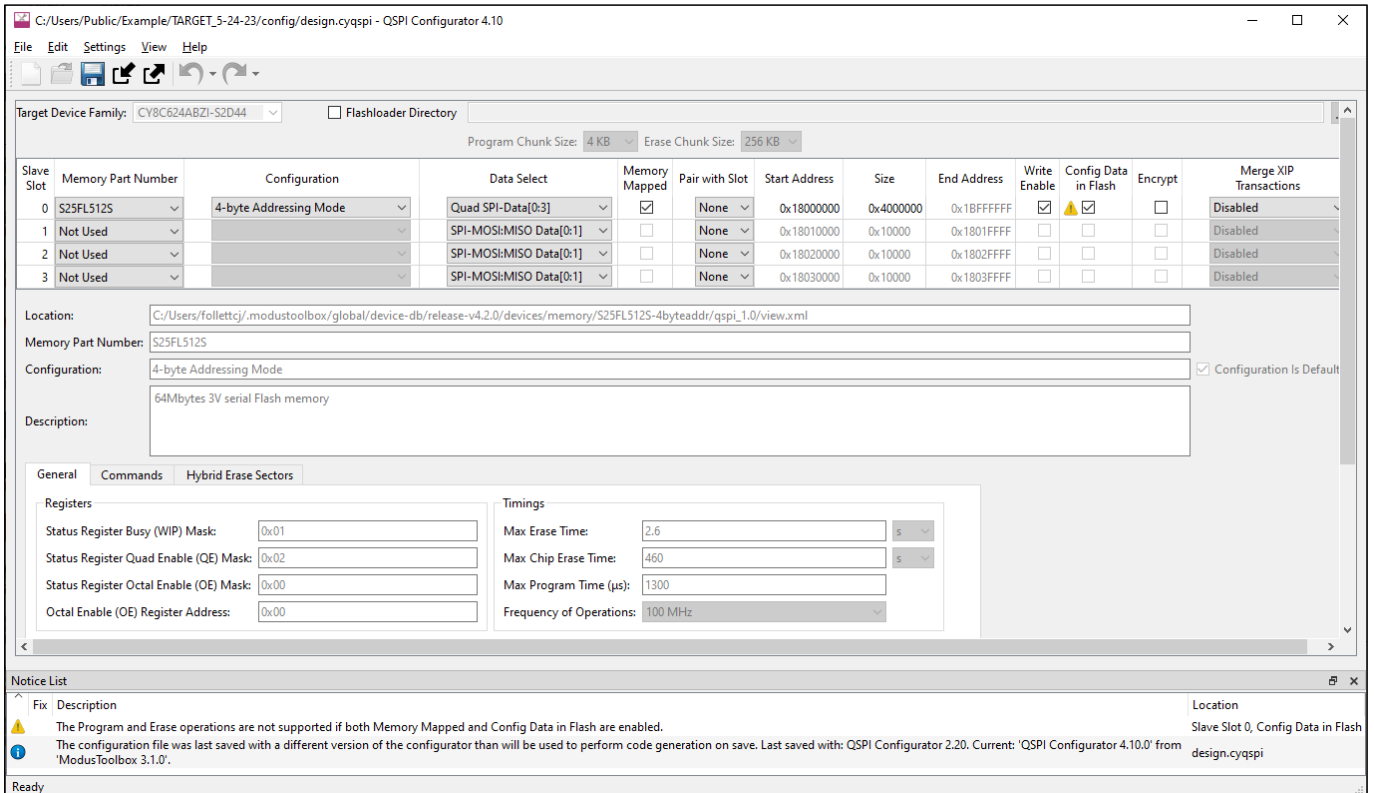
The QSPI Configurator uses and provides write access to two types of files: the QSPI configuration file (**.cyqspi*), and the memory configuration files: (*.cymem*). The QSPI Configurator uses the information from these files to generate the following files:

- **.cyqspi* – Controls the application-level settings used to provide access to the serial memory device. In the GUI, these settings are at the top of the main window. For the format and contents of the **.cyqspi* file, see [*.cyqspi schema](#).
- **.cymem* – Contains parameters of a specific serial memory device, including command formats, memory access sizes, and memory access timings. Unless you are creating a new **.cymem* file, these settings are typically read-only and are shown in the lower part of the main window. For the format and contents of **.cymem* files, see [*.cymem schema](#).
- *cy_qspi_memslot.c* – Contains structures populated with parameters specified by the **.cyqspi* file and the **.cymem* file or files.
- *cy_qspi_memslot.h* – Contains the declarations of the structures defined in *cy_qspi_memslot.c*.
- *qspi_config.cfg* – Contains a SMIF Bank layout for use with OpenOCD. It can be ignored if OpenOCD is not in use.

GUI description

5 GUI description

The top area of the QSPI Configurator is used for configuring memories; the read-only area below it displays information about the selected memory part number.



5.1 Menus

The QSPI Configurator contains the following menus.

5.1.1 File

- **New** – Creates a new Configuration file.
- **Open...** – Opens and loads an existing Configuration file, in either the current *.cyqspi* format or the obsolete *.cysmif* format.
- **Save** – Saves changes to the *.cyqspi* file. If a file has not been opened or is not in the *.cyqspi* format, the Save file dialog will open.
- **Save As...** – Saves changes to a new file.
- **Open in System Explorer** – This opens your computer’s file explorer tool to the folder that contains the Configuration file.
- **Import...** – Imports a specified configuration file.
- **Export...** – Exports the current configuration file into a specified file.
- **New *.cymem File...** – Creates a new memory file with default parameters. See [Create memory file](#).
- **Open *.cymem File...** – Opens an existing memory file.
- **Recent Files** – Shows up to five recent files that you can open directly.
- **Exit** – Closes the configurator.

GUI description

5.1.2 Edit

- **Undo** – Undoes the last action or sequence of actions.
- **Redo** – Redoes the last undone action or sequence of undone actions.

5.1.3 Settings

- **Use Memory Folder...** – Opens a dialog to set the default path for the memory file to be saved.
- **Local Content:** Check box to toggle between online and LCS mode, and read the local copy of the manifest file installed from the LCS Manager. Refer to the [LCS Manager CLI user guide](#) for more details.
- **Proxy Settings:** Opens a dialog to specify direct or manual proxy settings.

5.1.4 View

- **Notice List** – Shows/hides the Notice List pane, which contains any errors, warnings, tasks, and information notes. See the [Device Configurator user guide](#) for more details.
- **Toolbar** – Shows/hides the Toolbar.
- **Reset View** – Restores the Notice List and Toolbar to the default state.

5.1.5 Help

- **View Help** – Opens this document.
- **About QSPI Configurator** – Open the About box for version information, with links to open <https://www.infineon.com> and the current session log file.

5.2 Toolbar

The toolbar contains a few of the same commands included on the [File](#) and [Edit](#) menus.



5.3 Target Device Family

This pull-down menu shows the selected MPN for which the memory settings apply.

5.4 Flashloader Directory

A flashloader is an ELF file containing the algorithms and data structures required for flash memory programming. 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, as well as how the memory is configured. The patched files are placed in the *GeneratedSource/* directory alongside the source code and header files generated by the QSPI Configurator.

To enable flashloader patching, select the **Flashloader Directory** check box and use the [...] button to find and select the appropriate directory (or type the directory location in the text field). When you enable the **Flashloader Directory** check box, the following fields also become active to set appropriate chunk sizes:

- Program Chunk Size
- Erase Chunk Size

These settings allow you to control the granularity of the operations done while programming the flash memory. Larger values allow for more efficient data transfer, while smaller values can use memory more

GUI description

efficiently. When a memory slot is configured for Auto detect SFDP, the QSPI Configurator cannot know what values are legal, so ensure that these values are at least as large as the largest size of the program/erase operations of all the memories in use. Also, neither value can be larger than the smallest memory in use.

Note: You can also run `qspi-configurator` or `qspi-configurator-cli` via the command line using the `--flashloader-dir` option. If so, the next argument must be the directory path containing the flashloader(s) to patch. If you provide this argument on the command line, the flashloader directory cannot be changed in the GUI.

5.4.1 Patched flashloaders for AIROC™ CYW208xx devices

The QSPI Configurator stores files for patched flashloaders in the application directory. For Eclipse and VS Code, these files are handled automatically .

When exporting such applications to Keil μVision or IAR EWARM, these patched flashloader files must be copied into the appropriate 3rd party IDE directory. Refer to the [Keil μVision for ModusToolbox™ user guide](#) or the [IAR Embedded Workbench with ModusToolbox™ user guide](#) as applicable when exporting these applications.

5.5 QSPI configuration fields

The top part of the QSPI Configurator contains the following parameters:

Parameter	Description
Slave Slot	Specifies the slave slot being configured. This number corresponds to the slave select line that will be connected to the memory device.
Memory Part Number	Device part number represents the memory device that is being interfaced to the corresponding slave slot. You can select a memory device from the list, or select the option to auto detect the device. Based on the memory device selected, the corresponding *.cymem file is linked into the slave slot.
Configuration	For certain Memory Part Numbers, there is additional configuration information, such as 3-byte or 4-byte Addressing Mode.
Data Select	Allows you to select the data line options for the corresponding memory slot.
Memory Mapped	When this option is enabled, the configured memory device is mapped to the PSoC™ MCU's memory map. If disabled, access to memory must be done through the QSPI API.
Pair with Slot	Determines the paired slot for dual Quad operation.
Start Address	Determines the starting address where the memory device is going to be mapped in the PSoC™ memory map.
Size	Determines size in bytes of the memory device to be mapped in the PSoC™ memory map.
End Address	Represents the end address of the memory device as mapped in the PSoC™ MCU's memory map. This is a read-only field that is calculated automatically from "Start Address" and "Size" cells values.
Write Enable	This lets you enable or disable writes to the external memory in a memory mapped mode of operation.
Config Data in Flash	Determines whether a specific memory slot's config structures are to be placed in Flash or SRAM. When chosen to be placed in SRAM, the support for the programmer is not provided. Refer to AN228740 – Usage of Quad SPI (QSPI) Serial Memory Interface (SMIF) in PSoC™ 6 .
Encrypt	Determines whether to treat the memory device in the corresponding slave slot as an encrypted device. If the memory is mapped, all access to this memory will be decrypted on-the-fly. Setting this field does expect that the right encryption key is loaded as a part of the secure image.

GUI description

Parameter	Description
Merge XIP Transactions	Specifies how many cycles can pass between memory accesses while still skipping the overhead of re-sending the read command. For CAT1A (PSoC™ 6) devices, this field must be Disabled.

5.6 Edit memory file fields

The Edit Memory dialog contains the following fields. These fields also display as read-only in the lower part of the QSPI Configurator.

Field	Description
Location	The path and file name of the current memory file.
Memory Part Number	This is the name of the memory chip for which this configuration file is designed. This field will be displayed in the main QSPI Configurator window in the Memory Part Number drop-down menu.
Configuration	Some memory parts can be configured in more than one way, and each configuration is contained in a separate memory file. When creating more than one memory file for a part, use this field to describe how this file differs from the others. For example, "3-byte Addressing Mode" or "Hybrid Sectors at Bottom".
Configuration Is Default	For memory parts with more than one configuration, this check box specifies whether this configuration is the one that should be automatically selected when this memory part is chosen in the Memory Part Number drop-down menu.
Description	Blank field to type a description for the memory.

5.6.1 General tab

Field	Description
Status Register Busy (WIP) Mask	Mask for the busy bit in the status register.
Status Register Quad Enable (QE) Mask	Mask for the quad enable bit in the status register.
Status Register Octal Enable (OE) Mask	Mask for the octal enable bit in the status register.
Octal Enable (OE) Register Address	The address to use when Read Any Register / Write Any Register commands must be used to access the octal enable register.
Size of Memory (bytes)	Denotes the actual size of the memory device.
Program Page Size (bytes)	Denotes the page size for a program operation. This size provides the granularity with which program operations can be committed in the memory device.
Erase Block Size (bytes)	Provides the erase block size.
Number of Address Bytes for SPI Transactions	Sets the number of bytes that are expected for the address field in the QSPI transaction.
Max Erase Time	Time the device typically takes to erase a Erase Type 1 size. You must poll the device's busy status to determine whether the operation has completed. This field has no meaning if the corresponding Erase Type size is 00h.
Max Chip Erase Time	Typical time to erase one chip (die). You must poll the device's busy status to determine whether the operation has completed. For a device consisting of multiple dies that are individually accessed, the time is for each die to which a chip erase command is applied.

GUI description

Field	Description
Max Program Time (µs)	Typical time the device takes to write a full page. You must poll the device's busy status to determine whether the operation has completed. You may scale this by half or a quarter to determine approximate times for half and quarter page program operations.
Frequency of Operations	The frequency that the memory runs at in this configuration.
Latency Cycles Mask	Mask for the latency cycles bit or bits in the status register.
Latency Cycles Register Address	The address to use when Read Any Register / Write Any Register commands must be used to access the latency cycles register.

5.6.2 Commands tab

The **Commands** table tabs – **Single SPI, Dual SPI, Quad SPI, Octal SPI** – show the SPI data widths supported by the selected memory.

Field	Description
Description	List of commands: <ul style="list-style-type: none"> • Read • Write Enable • Write Disable • Erase Command • Chip Erase • Program • Read QE Register • Read WIP Register • Write QE Register • Read OE Register • Write OE Register • Read Latency • Write Latency
Number	Byte command word.
Command Width	Width of the command transfer.
Command Rate	Determines whether the command byte should be sent in Single Data Rate (SDR) or Double Data Rate (DDR) mode. CAT1A (PSoC™ 6) devices only support SDR.
Address Width	Width of the address transfer.
Address Rate	Determines whether the address bytes should be sent in SDR or DDR mode. CAT1A (PSoC™ 6) devices only support SDR.
Mode	Provides the mode word for the command.
Mode Width	Provides the width of the mode word transfer.
Mode Rate	Determines whether the mode byte should be sent in SDR or DDR mode. CAT1A (PSoC™ 6) devices only support SDR.
Dummy Cycles	Provides the number of dummy cycles in the transfer.
Data Width	Provides the width of data bytes in the transfer.
Data Rate	Determines whether the data bytes should be sent in SDR or DDR mode. CAT1A (PSoC™ 6) devices only support SDR.

GUI description

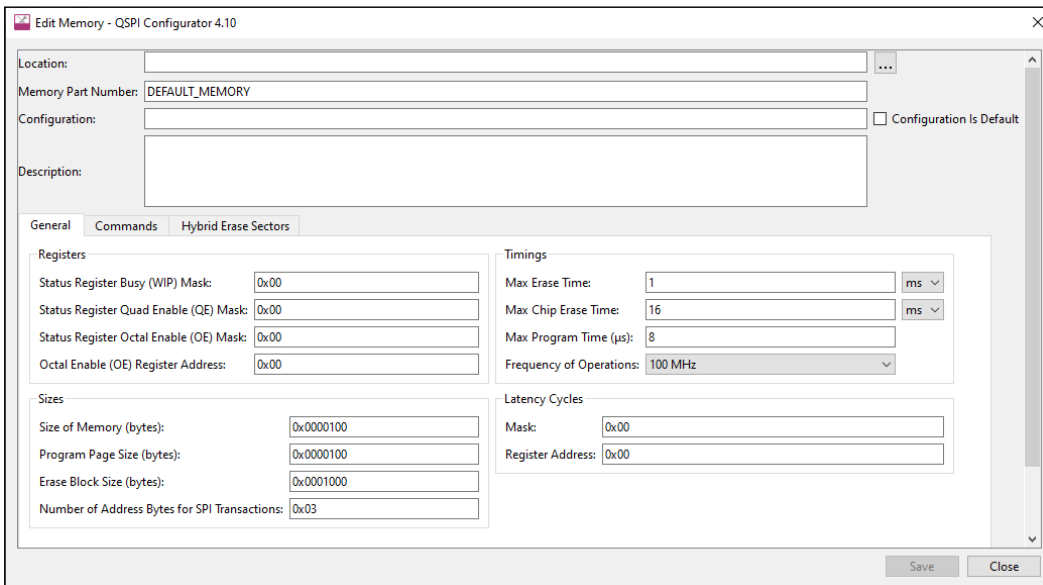
5.6.3 Hybrid Erase Sectors tab

Field	Description
Hybrid Erase Sectors	This check box specifies whether sectors of different sizes are present in this configuration. The table below contains details about those sectors. If this check box is not enabled, then there are no sectors with different sizes, and none of the other fields are active.
Row Buttons	Use these buttons to add and delete rows, as well as move rows up and down.
Region Table	<p>This table contains information about the various erase sector regions:</p> <ul style="list-style-type: none"> • The Region Start column specifies the start of the erase sector region. • The Sector Count column specifies how many erase sectors this region contains. • The Sector Size column specifies the size of the erase sectors in this region. • The Erase Command column specifies what command should be used to erase sectors in this region. • The Erase Time column specifies the maximum time that it can take to erase a sector in this region.

Create new memory file

6 Create new memory file

1. Select **New *.cymem File...** to open the Edit Memory window from the main QSPI Configurator:



2. Click [...] button next to the **Location** field to specify the file name and location of the memory configuration file (*.cymem). If you prefer, you can ignore the **Location** field for now, and then specify the file name and location when saving the file.
3. Enter a desired **Memory Part Number**. When selected, this field will be displayed in the main QSPI Configurator window in the **Memory Part Number**.
4. Complete the information for the remaining fields, as appropriate. See [Edit memory file fields](#).
5. Click **Save** to save the configured memory. If **Location** was not specified previously, this will open a save dialog to navigate to the appropriate location, type a file name, and click **Save**.
6. Click **Close** to return to the QSPI Configurator.

6.1 Memory database

The QSPI Configurator memory database is a set of default memory configurations, based on values from each memory’s datasheet. Check that the selected memory configuration is aligned with a particular part number.

Note: By default, some memory parts are configured with protected regions, which prevents the successful execution of program/erase memory commands.

Note: Dummy cycles may vary based on memory part configuration.

Note: The list of supported commands may vary between memory parts.

XML schemas

7 XML schemas

7.1 *.cyqspi schema

```

<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Configuration">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="DevicePath"/>
        <xs:element type="xs:string" name="FlashloaderDir"/>
        <xs:element type="xs:int" name="FlashloaderProgramChunkSize"/>
        <xs:element type="xs:int" name="FlashloaderEraseChunkSize"/>
        <xs:element name="SlotConfigs">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="SlotConfig" maxOccurs="4" minOccurs="0">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element type="xs:byte" name="SlaveSlot"/>
                    <xs:element type="xs:string" name="PartNumber"/>
                    <xs:element type="xs:boolean" name="MemoryMapped"/>
                    <xs:element type="xs:string" name="DualQuad"/>
                    <xs:element type="xs:string" name="StartAddress"/>
                    <xs:element type="xs:string" name="Size"/>
                    <xs:element type="xs:string" name="EndAddress"/>
                    <xs:element type="xs:boolean" name="WriteEnable"/>
                    <xs:element type="xs:boolean" name="Encrypt"/>
                    <xs:element type="xs:string" name="DataSelect"/>
                    <xs:element type="xs:string" name="MemoryConfigsPath"/>
                    <xs:element type="xs:boolean" name="ConfigDataInFlash"/>
                    <xs:element type="xs:string" name="MergeTimeout"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:attribute type="xs:string" name="app" fixed="QSPI"/>
  <xs:attribute type="xs:int" name="major"/>
  <xs:attribute type="xs:int" name="minor"/>
</xs:schema>

```

7.2 *.cymem schema

```

<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="CyMemoryConfiguration">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="PartNumber"/>
        <xs:element type="xs:string" name="Configuration" minOccurs="0"/>
        <xs:element type="xs:string" name="DisplayName" minOccurs="0"/>
        <xs:element type="xs:string" name="Description"/>
        <xs:element type="xs:string" name="NumberOfAddress"/>
        <xs:element type="xs:string" name="SizeOfMemory"/>
        <xs:element type="xs:string" name="EraseBlockSize"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

XML schemas

```

<xs:element type="xs:string" name="ProgramPageSize"/>
<xs:element type="xs:string" name="StatusRegisterBusyMask"/>
<xs:element type="xs:string" name="StatusRegisterQuadEnableMask"/>
<xs:element type="xs:int" name="EraseTime"/>
<xs:element type="xs:int" name="ChipEraseTime"/>
<xs:element type="xs:int" name="ProgramTime"/>
<xs:element name="HybridInfo" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="Region">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="RegionIndex" type="xs:int" />
            <xs:element name="SectorCount" type="xs:int" />
            <xs:element name="SectorSize" type="xs:string" />
            <xs:element name="EraseCommand" type="xs:string" />
            <xs:element name="EraseTime" type="xs:int" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="Commands" maxOccurs="4" minOccurs="1">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Command" maxOccurs="unbounded" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element type="xs:string" name="CommandDescription"/>
            <xs:element type="xs:string" name="CommandName"/>
            <xs:element type="xs:string" name="CommandNumber"/>
            <xs:element type="xs:string" name="CmdWidth"/>
            <xs:element type="xs:string" name="CmdRate"/>
            <xs:element type="xs:string" name="AddrWidth"/>
            <xs:element type="xs:string" name="AddrRate"/>
            <xs:element type="xs:string" name="Mode"/>
            <xs:element type="xs:string" name="ModeWidth"/>
            <xs:element type="xs:string" name="ModeRate"/>
            <xs:element type="xs:byte" name="DummyCycles"/>
            <xs:element type="xs:string" name="DataWidth"/>
            <xs:element type="xs:string" name="DataRate"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute type="xs:string" name="mode" use="optional"
default="Quad"/>
  </xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute type="xs:string" name="version" use="optional" default="1"/>
</xs:complexType>
</xs:element>
</xs:schema>

```


Version changes

8 Version changes

This section lists and describes the changes for each version of this tool.

Version	Change Descriptions
1.0	New tool.
1.1	Added Notice List and toolbar. Updated to accommodate back end changes.
2.0	Added Import/Export functionality. Moved configuration data from being embedded in the .h file to a new .cyqspi file. Set EraseTime, ChipEraseTime and ProgramTime to max. Allowed memory mapping for any Data Select selections. Added warning when "Config Data in Flash" check box isn't selected, and the device part number is not in "auto detect mode." Disabled the "Config Data in Flash" check box when the "Memory Part Number" is set to "Auto detect SFDP." Added handling of invalid command line arguments. Made it impossible for paired slots to have different memory part numbers. Fixed the Pair with Slot Memory Overlap Error.
2.1	Separated the Memory Part Number combo box into Memory Part Number and Configuration. Added support for hybrid erase sectors.
2.20	Added Undo/Redo commands. Added Copy feature to the Notice List.
2.30	Removed the command-line generate options: <code>-g</code> and <code>--generate</code> . Removed version 1.1 personality support.
3.0	Added support for flashloader patching, and added the Flashloader directory field. Removed support for loading configuration data from .h files. Configuration information is now always located in the <i>design.cyqspi</i> file.
4.0	Changed the device library file from xml to <i>props.json</i> . Updated the GUI layout. Added the ability to load memory configuration from device-db.
4.10	Added Settings menu. Updated the memory file GUI and added several new fields: <ul style="list-style-type: none"> • Status Register Octal Enable (OE) Mask • Octal Enable (OE) Register Address • Frequency of Operations • Latency Cycles Mask • Latency Cycles Register Address

Revision history

Revision history

Date	Revision	Description
**	2018-11-27	New document.
*A	2019-02-27	Added Notice List and toolbar. Updated to accommodate back end changes.
*B	2019-10-16	Updated to version 2.0.
*C	2020-03-27	Updated to version 2.1.
*D	2020-04-21	Updated text for hybrid memory descriptions.
*E	2020-08-28	Updated to version 2.20.
*F	2020-09-07	Updated to mention the UNITS (bytes/word) of size explicitly. Updated description for "End Address" field. Updated description for "Start address" field. Added a link to App Note AN228740 for the "Config Data in Flash" field.
*G	2021-03-12	Updated to version 2.30.
*H	2022-01-20	Updated to version 3.0.
*I	2022-09-28	Updated to version 4.0.
*J	2023-05-24	Updated to version 4.10. Added references to 3 rd party IDE user guides.

Trademarks

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

Edition 2023-05-24

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2023 Infineon Technologies AG.

All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

002-24376 Rev. *J

Important notice

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffungsgarantie")

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

Warnings

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.