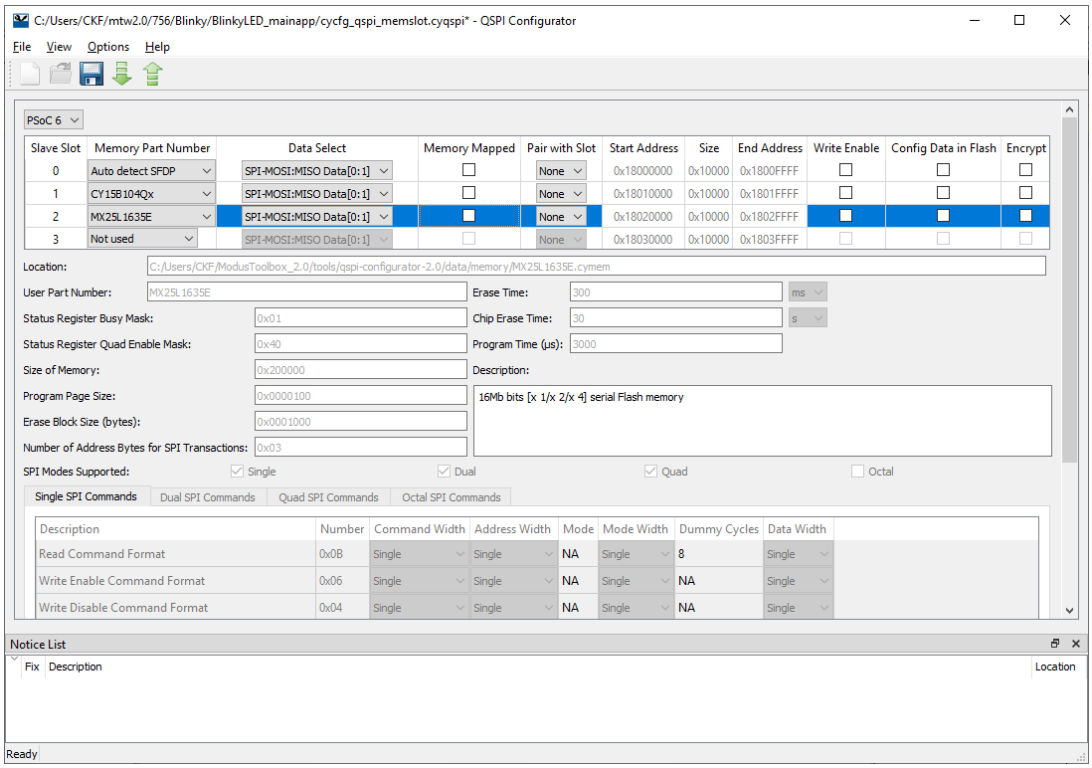
ModusToolbox™ QSPI Configurator Guide

Version 2.0

Overview

The Quad Serial Peripheral Interface (QSPI) Configurator is part of a collection of tools included in the ModusToolbox software. Use the QSPI Configurator to open or create configuration files, configure memory slots, and generate code for your application. The QSPI Configurator is a stand-alone tool. Use the top area for configuring memories; the read-only area below it displays information about the selected memory part number.



Supported Software

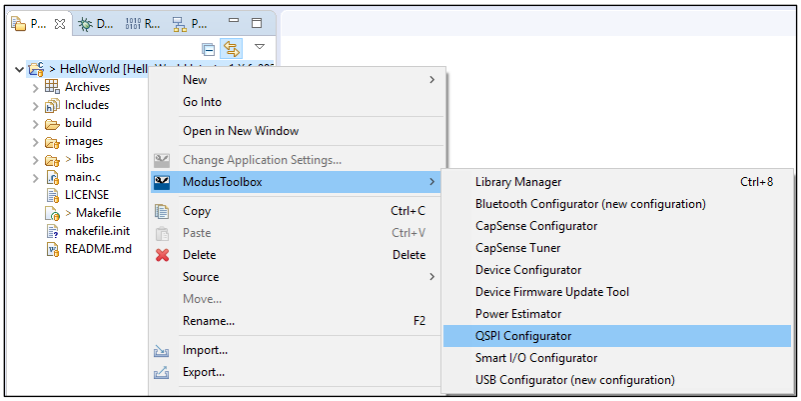
Name	Version	Link
Cypress PSoC 6 Peripheral Driver Library	1.3.0	https://github.com/cypresssemiconductorco/psoc6pdl

Launch the Configurator

You can run the QSPI Configurator from, and use it with, a ModusToolbox IDE application. You can also run it independently of the IDE. Then, you can either use the generated source with a ModusToolbox IDE application, or use it in any software environment you choose.

From a ModusToolbox IDE Application

If your ModusToolbox IDE application includes a *design.cyqspi* file in the Board Support Package (BSP), right-click on a project in the IDE Project Explorer, and select **ModusToolbox Configurators > QSPI Configurator** to open the tool.

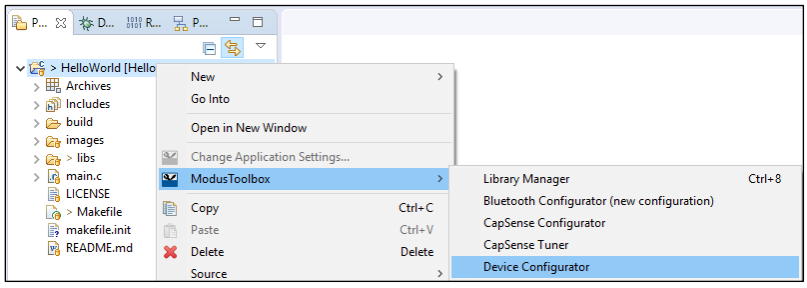


Note You can also launch the QSPI Configurator from a link in the Quick Panel.

From the Device Configurator

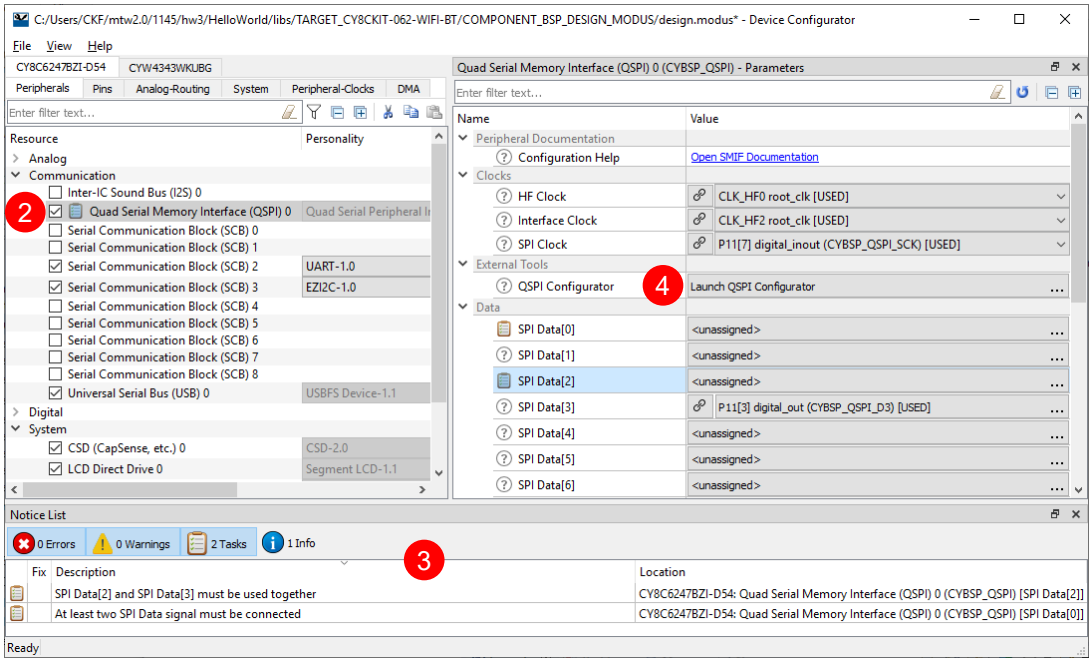
If your ModusToolbox IDE application does **not** include a *design.cyqspi* file:

1. Open the Device Configurator.



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.

Refer to the *Device Configurator Guide* for more information.



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.

Independent of the ModusToolbox IDE

To run the QSPI Configurator independently, navigate to the install location and run the executable. The default install location for the QSPI Configurator on Windows is:

<install_dir>\tools_<version>\qspi-configurator

For other operating systems, the installation directory will vary, based on how the software was installed.

When run independently, 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.

- If you open a Configuration file from a ModusToolbox IDE application, it will be the same flow as if you opened it from the ModusToolbox IDE.
- If you open a Configuration file from a non-ModusToolbox IDE application, the flow will be for your preferred working environment.
- If you create a new Configuration file, specify the file name and location to store the file.

From the Command Line

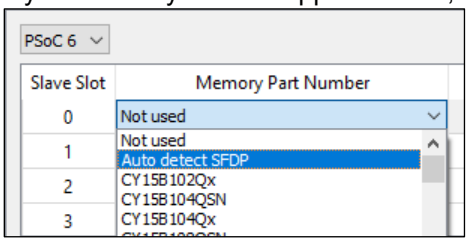
You can run the configurator from the command line. However, there are only a few reasons to do this in practice. The primary use case would be to re-generate source code based on the latest configuration settings. This would often be part of an overall build script for the entire application.

For information about command line options, run the configurator using the `-h` option.

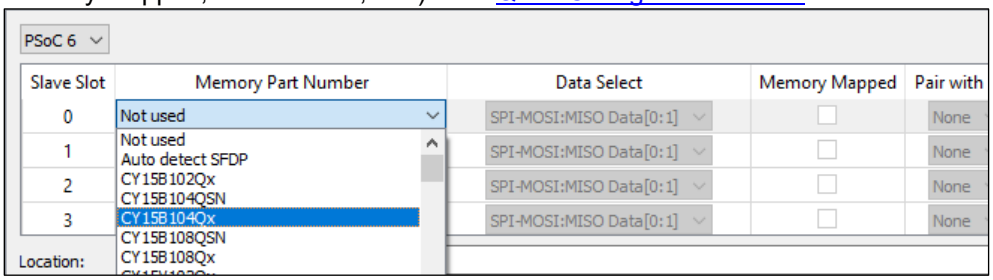
Quick Start

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

1. Launch the configurator.
2. Select a memory from the **Memory Part Number** list.
   a. If your memory device supports SFDP, select **Auto detect SFDP**.



   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.



   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.
3. Save the Configuration file to generate source code. See Code Generation.
4. Use the generated structures as input parameters for QSPI functions in your application.

Code Generation

The QSPI Configurator generates code into a “GeneratedSource” directory in your BSP, or in the same location you saved the Configuration file for non-ModusToolbox 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.

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 *cycfg_qspi_memslot.c*, *cycfg_qspi_memslot.h*, and *qspi_config.cfg* files.

- **.cyqspi* – This file 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* – These files contain 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* – This file is generated by the QSPI Configurator and contains structures populated with parameters specified by the **.cyqspi* file and the **.cymem* file or files.
- *cy_qspi_memslot.h* – This file is generated by the QSPI Configurator and contains the declarations of the structures defined in *cy_qspi_memslot.c*.
- *qspi_config.cfg* – This file contains a SMIF Bank layout for use with OpenOCD. It can be ignored if OpenOCD is not in use.

QSPI Configurator Menus

The QSPI Configurator contains the following menus.

File

- **New:** Create a new Configuration file.
- **Open…:** Open and load an existing Configuration file, in either the current .cyqspi format or the obsolete formats (.cysmif or embedded inside a .h file).
- **Save:** Save 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…:** Save changes to a new file.
- **Import…:** Imports a specified configuration file.
- **Export…:** Exports the current configuration file into a specified file.
- **New *.cymem File…:** Create a new memory file with default parameters. See Create Memory File.
- **Open *.cymem File…:** Open an existing memory file.
- **Recent Files:** Shows up to five recent files that you can open directly.
- **Exit:** Close the configurator.

View

- **Notice List:** Shows/hides the Notice List pane, which contains any errors, warnings, tasks, and information notes. See the *Device Configurator Guide* for more details.
- **Toolbar:** Shows/hides the Toolbar.
- **Reset View:** Restores the Notice List and Toolbar to the default state.

Options

- **Settings…:** Opens the Settings dialog to set the default path for the memory file to be saved.

Help

- **View Help:** Opens this document.
- **About QSPI Configurator:** Open the About box for version information.

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.
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. This field is a representation of the size of the memory being mapped.
Size	Determines size of the memory device to be mapped in the PSoC memory map. This field is a representation of the size of the memory being mapped.
End Address	Represents the end address of the memory device as mapped in the PSoC MCU's memory map. This field is a representation of the size of the memory being mapped. It is calculated automatically from "Start Address" and "Size" cells values.
Write Enable	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.
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.

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	Path and file name of the current memory file.
User Part Number	User-defined field for the part. This field will be displayed in the main QSPI Configurator window in the **Memory Part Number**, when it is selected.
Status Register Busy Mask	Mask for the busy bit in the status register.
Status Register Quad Enable Mask	Mask for the quad enable bit in the status register.
Size of Memory	Denotes the actual size of the memory device.
Program Page Size	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	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.

Field	Description
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.
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.
Program Time	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.
Description	Blank field to type a description for the memory.
SPI Modes Supported	This shows the SPI data widths supported by the selected memory.
Lower Table	
Description	List of commands: • Read Command Format • Write Enable Command Format • Write Disable Command Format • Erase Command Format • Chip Erase Command Format • Program Command Format • Read Status Register Command (containing Quad Enable bit) • Read Status Register Command (containing Work In Progress / Busy bit) • Write Status Register Command (containing Quad Enable bit)
Number	Byte command word.
Command Width	Width of the command transfer.
Address Width	Width of the address transfer.
Mode	Provides the mode word for the command.
Mode Width	Provides the width of the mode word transfer.
Dummy Cycles	Provides the number of dummy cycles in the transfer.
Data Width	Provides the width of data bytes in the transfer.

Memory Database

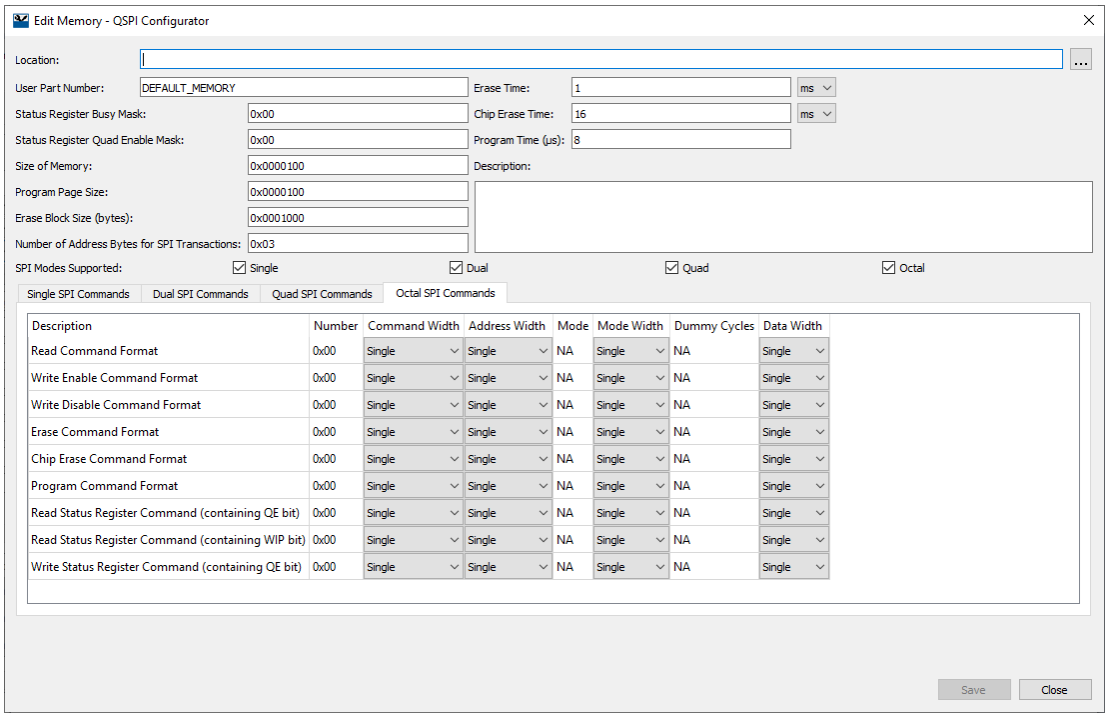
The QSPI Configurator memory database is a set of default memory configurations, based on values from each memory's datasheet. Check that the selected memory configuration is aligned with a particular part number.

Notes:

- By default, some memory parts are configured with protected regions, which prevents the successful execution of program/erase memory commands.
- Dummy cycles may vary based on memory part configuration.
- The list of supported commands may vary between memory parts.

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 **User Part Number**. This field will be displayed in the main QSPI Configurator window in the **Memory Part Number**, when it is selected.
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. Clock **Close** to return to the QSPI Configurator.

Migration of Configuration File Format

Versions of the QSPI Configurator prior to 2.0 used a C header file to store its configuration as a comment in XML format. In version 2.0, the configuration is stored in a separate *.cyqspi* file in XML format. Use the following instructions to migrate to the *.cyqspi* file as appropriate:

From the ModusToolbox IDE

1. Launch the QSPI Configurator. If there is no *.cyqspi* file, and if the .h file contains an XML configuration, it will be loaded automatically.
2. Click **Save** to create the *.cyqspi* file and update the C file.

Independent of the ModusToolbox IDE

1. Launch the QSPI Configurator.
2. Click **Open**.

3. Next to the **File name**, select the **Obsolete configurator files (*.h)** option and choose a header file.
4. After the configuration loads, click **Save** to create the .cyqspi file and update the C file.

Notes

- The .cyqspi file is located one directory up related to the header file.
- The command-line argument `--config` does not accept obsolete configuration files (*.h).

XML Schemas

*.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 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:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute type="xs:string" name="app" fixed="QSPI"/>
      <xs:attribute type="xs:int" name="major"/>
      <xs:attribute type="xs:int" name="minor"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

*.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="DisplayName"/>
        <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: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="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="AddrWidth"/>
                    <xs:element type="xs:string" name="Mode"/>
                    <xs:element type="xs:string" name="ModeWidth"/>
                    <xs:element type="xs:byte" name="DummyCycles"/>
                    <xs:element type="xs:string" name="DataWidth"/>
                  </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:complexType>
  </xs:element>
</xs:schema>
```

References

Refer to the following documents for more information, as needed:

- Device Configurator Guide
- ModusToolbox IDE User Guide
- ModusToolbox Release Notes

- PDL Reference Guide
- Device Datasheets
- Device Technical Reference Manuals

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.