



SMIF Configuration Tool 1.0

User Guide

Doc. No. 002-19118 Rev. *B

Cypress Semiconductor
198 Champion Court
<http://www.cypress.com>

© Cypress Semiconductor Corporation, 2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.

Contents



1. Introduction.....	4
1.1 Installation	4
1.2 Menus.....	5
1.2.1 File.....	5
1.2.2 Run	5
1.2.3 Options	5
1.2.4 Help	5
1.3 Supported Drivers.....	5
1.4 Memory Database	5
1.5 Product Upgrades.....	6
1.6 Support.....	6
1.7 Document Conventions	6
2. Quick Start	7
2.1 Invoking from PSoC Creator.....	7
2.2 Opening SMIF Configuration Tool Directly	9
2.2.1 Create/Open Memory Configuration	10
2.3 Using SMIF Generation Tool	10
2.4 Error Handling	10
3. Example.....	11
4. XML Description	18
4.1 *.cysmif File	18
4.2 *.cymem File.....	19
4.3 GUI, XML Parameters, and C-Structures	20
4.3.1 Slot Parameters (*.cysmif file)	20
4.3.2 Memory Parameters (*.cymem file)	21
4.3.3 Command Parameters (*.cymem file).....	22
4.3.4 Command List (*.cymem file).....	22
5. Error Handling	23
Revision History.....	24

1. Introduction



The SMIF Configuration Tool version 1.0 consists of two applications: SMIF Configuration GUI Tool and the command-line SMIF Generation Tool (*cysmifcodegen*). This tool allows you to generate input structures for the SMIF driver memslot API.

The cross-platform SMIF Generation Tool (*cysmifcodegen*) can be run under Windows, Linux, or Mac OS.

Slave slot	Memory part number	Data select	Memory mapped	Pair with slot	Start address	Size	End address	Write enable	Config Data In Flash	Encrypt
0	Not used	SPI-MOSI:MISO Data[0:1]	<input type="checkbox"/>	None	0x18000000	0x10000	0x18010000	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
1	Not used	SPI-MOSI:MISO Data[0:1]	<input type="checkbox"/>	None	0x18010001	0x25B30B	0x1826B3...	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	Not used	SPI-MOSI:MISO Data[0:1]	<input type="checkbox"/>	None	0x1826B3...	0x936D3	0x182FE9...	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	Not used	SPI-MOSI:MISO Data[0:1]	<input type="checkbox"/>	None	0x182FE9...	0x294BF	0x18327E...	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Location:

User part number:

Erase time: ms

Status register busy mask:

Chip erase time: ms

Status register quad enable mask:

Program time (us):

Size of memory:

Description:

Program page size:

Erase block size (bytes):

Number of address bytes for SMIF transactions:

Description	Number	Command width	Address width	Mode	Mode width	Dummy cycles	Data width
Read command format	0x00	Single	Single	NA	Single	NA	Single
Write enable command format	0x00	Single	Single	NA	Single	NA	Single
Write disable command format	0x00	Single	Single	NA	Single	NA	Single
Erase command format	0x00	Single	Single	NA	Single	NA	Single
Chip erase command format	0x00	Single	Single	NA	Single	NA	Single
Program command format	0x00	Single	Single	NA	Single	NA	Single
Read status register command (containing QE bit)	0x00	Single	Single	NA	Single	NA	Single
Read status register command (containing WIP bit)	0x00	Single	Single	NA	Single	NA	Single
Write status register command (containing QE bit)	0x00	Single	Single	NA	Single	NA	Single

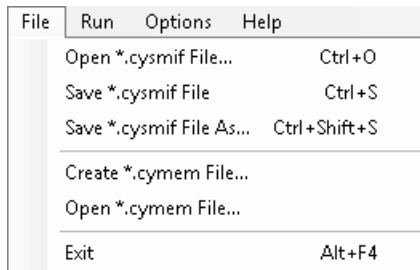
1.1 Installation

The SMIF Configuration Tool is bundled with the Peripheral Driver Library (PDL) version 3.0.1 or later, so you must install the PDL to use the SMIF Configuration Tool. See "Cypress Peripheral Driver Library v3.0 Quick Start Guide" for details.

1.2 Menus

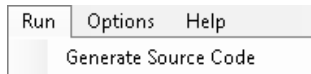
The SMIF Configuration Tool contains the following menus.

1.2.1 File



- **Open *.cysmif File:** Open and load an existing *.cysmif file.
- **Save *.cysmif File:** Save changes to the file. If the file does not exist, the Save file dialog will open.
- **Save *.cysmif File As...:** Save changes to a new file.
- **Create *.cymem File:** Create a new memory file with default parameters.
- **Open *.cymem File:** Open an existing memory file.
- **Exit:** Close the application.

1.2.2 Run

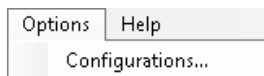


- **Generate Source Code:** Generates *.c and *.h files based on the configuration set in the *.cysmif file.

The location of the generated source is set by the configuration window <Output folder> value. If the output folder value is not selected, you can select the folder when you generate the source code.

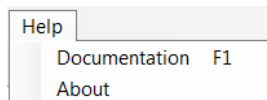
In addition, when the SMIF Configuration Tool is invoked from PSoC Creator, the code generation step is built in as a part of the build and compile process.

1.2.3 Options



- **Configurations:** Opens the configuration window.

1.2.4 Help



- **Documentation:** Opens this user guide.
- **About:** Open the About box for version information.

1.3 Supported Drivers

- SMIF driver v1.0.0

1.4 Memory Database

The SMIF Configuration Tool memory database is a set of default memory configurations, based on values from each memory's datasheet. The user must check that the selected memory configuration is aligned with a particular part number. Please note:

- By default, the memory database is configured for QuadSPI mode only.
- 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.

1.5 Product Upgrades

Cypress provides scheduled upgrades and version enhancements for PSoC Creator and the SMIF Configuration Tool, free-of-charge. You can download them directly from www.cypress.com under **Support & Community > Software Tools**.

In addition, critical updates to system documentation are provided under **Design Resources**.

1.6 Support

Free support for the SMIF Configuration Tool is available online. If PSoC Creator and the SMIF Configuration Tool are installed on your system, you can find the version, build, and/or service pack information from **Help > About**.

Visit <http://www.cypress.com/support> for online technical support. The resources include:

- Training Seminars
- Discussion Forums
- Application Notes
- Developer Community
- Knowledge Base
- Technical Support

You can also view and participate in discussion threads about a wide variety of device topics.

1.7 Document Conventions

The following table lists the conventions used throughout this guide:

Convention	Usage
Courier New	Displays file locations, user entered text, and source code: <code>C:\ ..cd\icc\</code>
<i>Italics</i>	Displays file names and reference documentation: <i>sourcefile.hex</i>
Bold	Displays keyboard commands in procedures: Enter or Ctrl + C
File > New Project	Represents menu paths: File > New Project > Clone
Bold	Displays commands, menu paths and selections, and icon names in procedures: Click the Debugger icon, and then click Next .

2. Quick Start

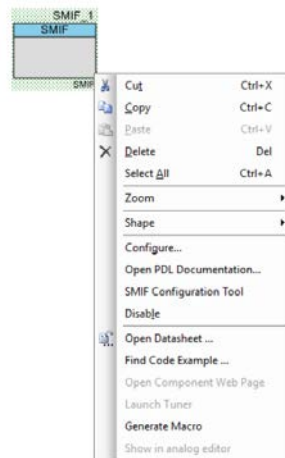


This chapter describes how to use the SMIF Configuration Tool for the most common use case: selecting a memory module and generating configuration code.

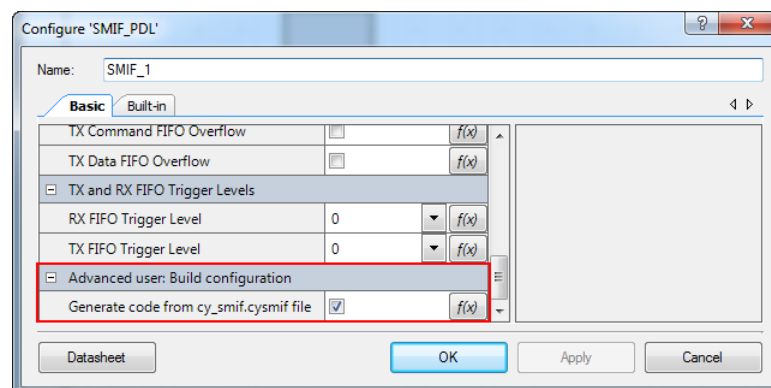
- [Invoking from PSoC Creator](#)
- [Opening SMIF Configuration Tool Directly](#)
- [Using SMIF Generation Tool](#)

2.1 Invoking from PSoC Creator

1. Right-click on the SMIF Component instance, and select **SMIF Configuration Tool**.

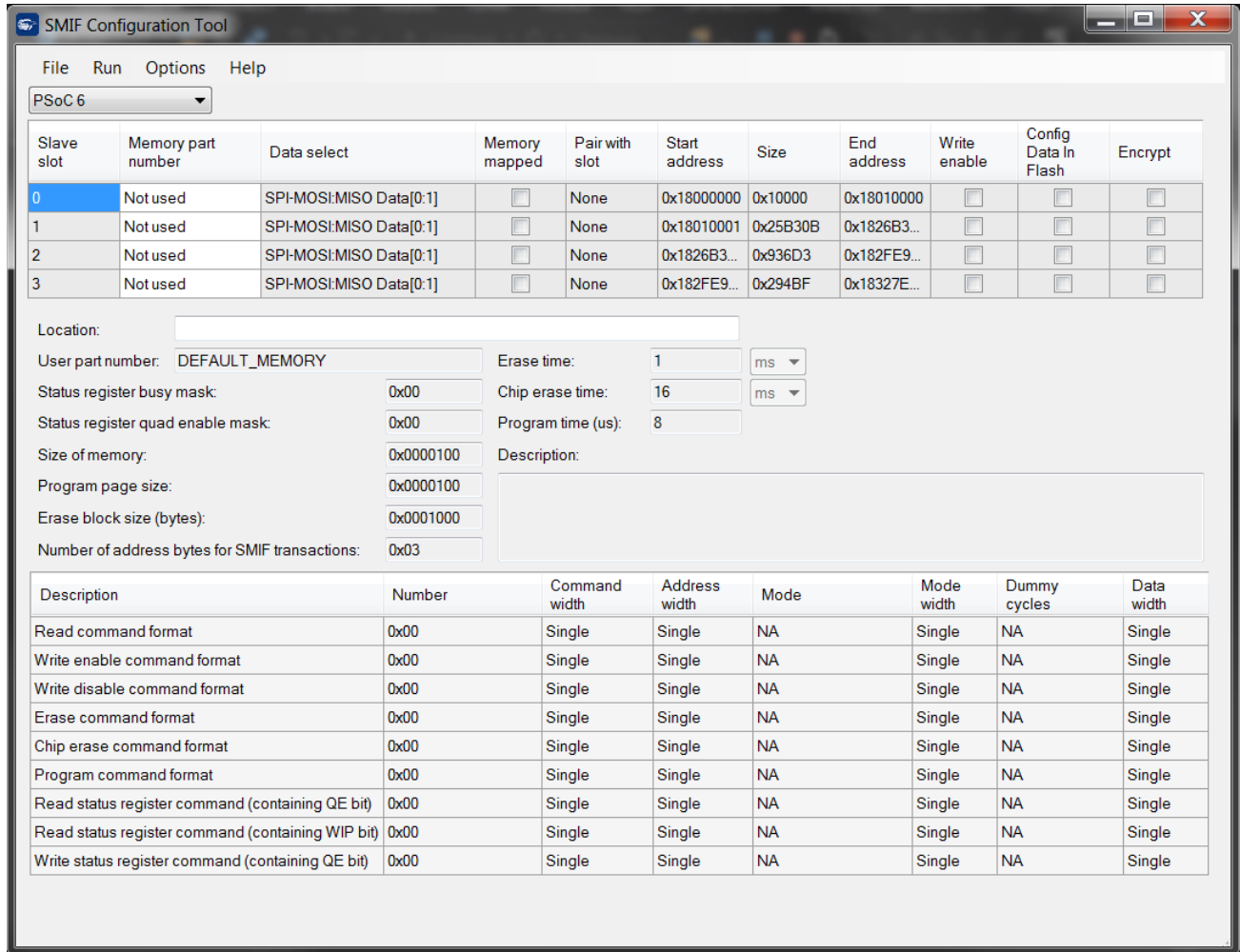


Note: If the SMIF Configuration Tool menu item is disabled, then code generation from the **.cysmif* file is disabled. To enable code generation, open PSoC Creator and double-click the SMIF Component to open the **Configure 'SMIF_PDL'** dialog. At the end of the list of options, select the **Generate code from cy_smif.cysmif file** check box.



Note: Once opened from within PSoC Creator, do not use menu items **File > Open Smif**, **File > Save AS Smif**, **Run**, or **Options > Output folder**. These items are used only if SMIF Configuration Tool is run as standalone application.

- On the SMIF Configuration Tool, select a memory from the **Memory part number** list, and specify the configuration parameters if required (for example, memory mapped, write enable).



Slave slot	Memory part number	Data select	Memory mapped	Pair with slot	Start address	Size	End address	Write enable	Config Data In Flash	Encrypt
0	Not used	SPI-MOSI:MISO Data[0:1]	<input type="checkbox"/>	None	0x18000000	0x10000	0x18010000	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
1	Not used	SPI-MOSI:MISO Data[0:1]	<input type="checkbox"/>	None	0x18010001	0x25B30B	0x1826B3...	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	Not used	SPI-MOSI:MISO Data[0:1]	<input type="checkbox"/>	None	0x1826B3...	0x936D3	0x182FE9...	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	Not used	SPI-MOSI:MISO Data[0:1]	<input type="checkbox"/>	None	0x182FE9...	0x294BF	0x18327E...	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Location:

User part number:

Erase time:

Status register busy mask:

Chip erase time:

Status register quad enable mask:

Program time (us):

Size of memory:

Description:

Program page size:

Erase block size (bytes):

Number of address bytes for SMIF transactions:

Description	Number	Command width	Address width	Mode	Mode width	Dummy cycles	Data width
Read command format	0x00	Single	Single	NA	Single	NA	Single
Write enable command format	0x00	Single	Single	NA	Single	NA	Single
Write disable command format	0x00	Single	Single	NA	Single	NA	Single
Erase command format	0x00	Single	Single	NA	Single	NA	Single
Chip erase command format	0x00	Single	Single	NA	Single	NA	Single
Program command format	0x00	Single	Single	NA	Single	NA	Single
Read status register command (containing QE bit)	0x00	Single	Single	NA	Single	NA	Single
Read status register command (containing WIP bit)	0x00	Single	Single	NA	Single	NA	Single
Write status register command (containing QE bit)	0x00	Single	Single	NA	Single	NA	Single

Note: If the required memory is not present in the list, you can add it. See [Create/Open Memory Configuration](#).

- Save the configured data into the *.cysmif file.

Note: The *.cysmif file must be saved in the project root folder.

- Build the project in PSoC Creator.

The build process invokes the command-line code generator. If the **Generate code from cy_smif.cysmif file** check box is not selected (see [Step 1](#)), then the command line tool will not run.

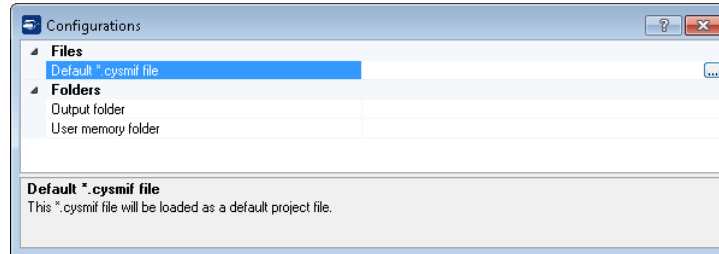
If successful, generated *.h and *.c files are added to the project.

- Use the generated structures as input parameters for SMIF functions.

2.2 Opening SMIF Configuration Tool Directly

1. Run the *SMIFConfigurationTool.exe* file.

The default *.cysmif file is loaded. The path to this file is set in **Option > Configurations > Files > Default *.cysmif file**.

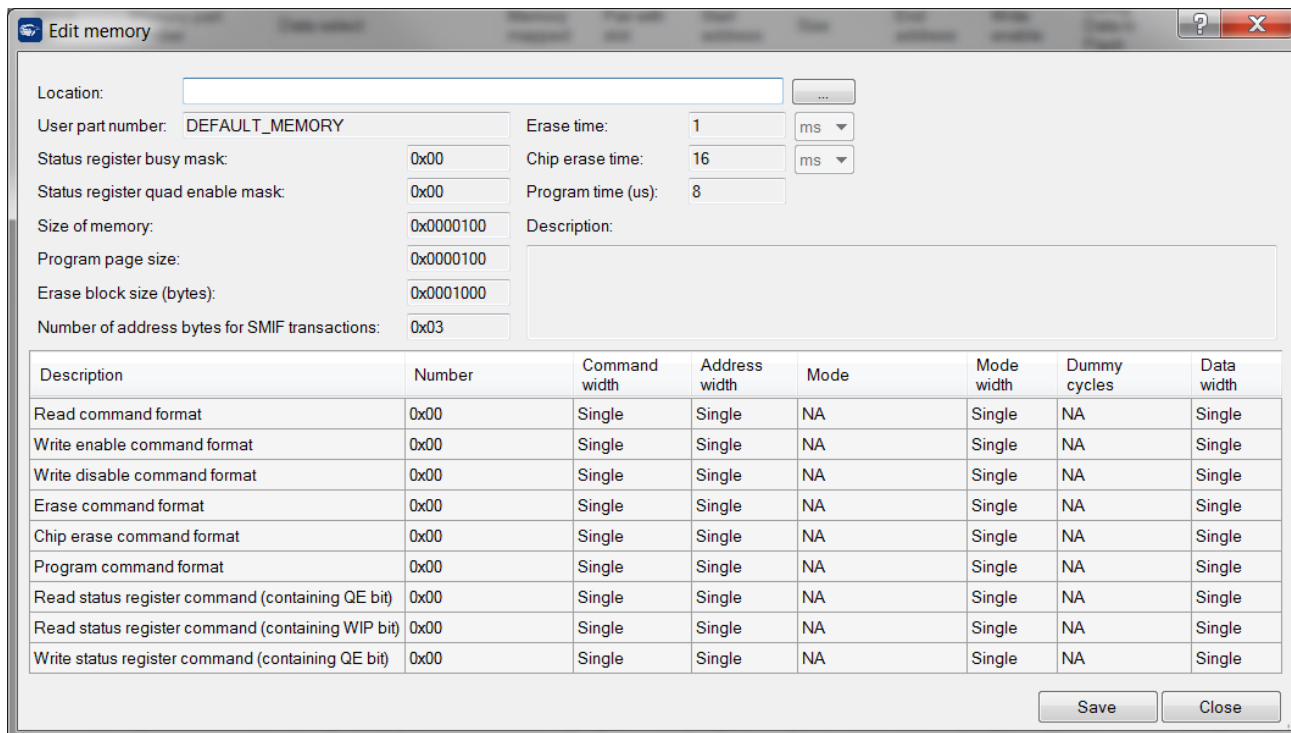


If the path is not set, parameters of all slots are set with default values.

- **<Default *.cysmif file>**: The *.cysmif file that is loaded when the tool is started.
 - **<Output folder>**: The output folder for generated source code files.
 - **<User memory folder>**: The folder where *.cymem files are located.
2. In the SMIF Configuration Tool, select memory from the **Memory part number** list, and specify the configuration parameters such as memory mapped and write enable if required.
Note: If the required memory is not present in list, you can add it. See [Create/Open Memory Configuration](#).
 3. Save the configured data into the *.cysmif file.
Note: The *.cysmif file must be saved in the project root folder.
 4. When all parameters are configured and the *.cysmif file is saved, you can invoke the command-line tool to generate *.h and *.c files into the output folder. See [Using SMIF Generation Tool](#).

2.2.1 Create/Open Memory Configuration

1. To configure memories, open the Edit memory configuration window (**File > Create *.cymem File** or **File > Open *.cymem File**).



Description	Number	Command width	Address width	Mode	Mode width	Dummy cycles	Data width
Read command format	0x00	Single	Single	NA	Single	NA	Single
Write enable command format	0x00	Single	Single	NA	Single	NA	Single
Write disable command format	0x00	Single	Single	NA	Single	NA	Single
Erase command format	0x00	Single	Single	NA	Single	NA	Single
Chip erase command format	0x00	Single	Single	NA	Single	NA	Single
Program command format	0x00	Single	Single	NA	Single	NA	Single
Read status register command (containing QE bit)	0x00	Single	Single	NA	Single	NA	Single
Read status register command (containing WIP bit)	0x00	Single	Single	NA	Single	NA	Single
Write status register command (containing QE bit)	0x00	Single	Single	NA	Single	NA	Single

If you selected the **File > Create *.cymem File** option, all parameters are filled with default values. Default memory files cannot be changed.

- Use **Save** to save changes and close the window.
- Use **Close** to close the window and discard unsaved changes.

2.3 Using SMIF Generation Tool

Run the *cysmifcodegen.exe* file from the command line as follows:

Windows:

```
cysmifcodegen.exe -generate <cysmif_path> -output <output_folder>
```

Linux and macOS:

```
./cysmifcodegen -generate <cysmif_path> -output <output_folder>
```

- The saved **.cysmif* file is passed as the first argument to the command-line tool.
- The second argument is the output folder specified under **Configuration > Folders > Output folder**.

2.4 Error Handling

If an error occurs, only a **.h* file will be generated with the following contents:

```
#error <Error message>
```

3. Example



- 1) Run the SMIF Configuration Tool (See “Quick Start” section for details).
- 2) Configure one or several slots. For example, set these parameters:
 - a) “Memory part number” – MX25L12835F
 - b) check “Memory mapped” check boxLeave other parameters unchanged.
- 3) Save *.cysmif file: **File > Save *.cysmif File**
- 4) The .cysmif file is created. The contents of the file look like this:

```
<?xml version="1.0" encoding="utf-16"?>
<CySMIFConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <DevicePath>devices\PSoc_6.xml</DevicePath>
  <SlotConfigs>
    <SlotConfig>
      <SlaveSlot>0</SlaveSlot>
      <PartNumber>MX25L12835F</PartNumber>
      <MemoryMapped>true</MemoryMapped>
      <DualQuad>None</DualQuad>
      <StartAddress>0x18000000</StartAddress>
      <Size>0x10000</Size>
      <EndAddress>0x18010000</EndAddress>
      <WriteEnable>false</WriteEnable>
      <Encrypt>false</Encrypt>
      <DataSelect>SPI_MOSI_MISO_DATA_0_1</DataSelect>
      <Description />
      <MemoryConfigsPath>MX25L12835F</MemoryConfigsPath>
      <ConfigDataInFlash>false</ConfigDataInFlash>
    </SlotConfig>
    <SlotConfig>
      <SlaveSlot>1</SlaveSlot>
      <PartNumber>Not used</PartNumber>
      <MemoryMapped>false</MemoryMapped>
      <DualQuad>None</DualQuad>
      <StartAddress>0x18010001</StartAddress>
      <Size>0x25B30B</Size>
      <EndAddress>0x1826B30C</EndAddress>
      <WriteEnable>false</WriteEnable>
      <Encrypt>false</Encrypt>
      <DataSelect>SPI_MOSI_MISO_DATA_0_1</DataSelect>
      <Description />
    </SlotConfig>
    <SlotConfig>
      <SlaveSlot>2</SlaveSlot>
      <PartNumber>Not used</PartNumber>
      <MemoryMapped>false</MemoryMapped>
    </SlotConfig>
  </SlotConfigs>
  <MemoryConfigsPath>D:\cypress\projects\SMIF\source\GIT\SMIFConfigurationTool\smif\cysmifc
onfigtool\bin\Release\templates\default_memory.xml</MemoryConfigsPath>
  <ConfigDataInFlash>false</ConfigDataInFlash>
</CySMIFConfiguration>
```

```

        <DualQuad>None</DualQuad>
        <StartAddress>0x1826B30D</StartAddress>
        <Size>0x936D3</Size>
        <EndAddress>0x182FE9E0</EndAddress>
        <WriteEnable>>false</WriteEnable>
        <Encrypt>>false</Encrypt>
        <DataSelect>SPI_MOSI_MISO_DATA_0_1</DataSelect>
        <Description />

<MemoryConfigsPath>D:\cypress\projects\SMIF\source\GIT\SMIFConfigurationTool\smif\cysmifc
onfigtool\bin\Release\templates\default_memory.xml</MemoryConfigsPath>
    <ConfigDataInFlash>>false</ConfigDataInFlash>
</SlotConfig>
<SlotConfig>
    <SlaveSlot>3</SlaveSlot>
    <PartNumber>Not used</PartNumber>
    <MemoryMapped>>false</MemoryMapped>
    <DualQuad>None</DualQuad>
    <StartAddress>0x182FE9E1</StartAddress>
    <Size>0x294BF</Size>
    <EndAddress>0x18327EA0</EndAddress>
    <WriteEnable>>false</WriteEnable>
    <Encrypt>>false</Encrypt>
    <DataSelect>SPI_MOSI_MISO_DATA_0_1</DataSelect>
    <Description />

<MemoryConfigsPath>D:\cypress\projects\SMIF\source\GIT\SMIFConfigurationTool\smif\cysmifc
onfigtool\bin\Release\templates\default_memory.xml</MemoryConfigsPath>
    <ConfigDataInFlash>>false</ConfigDataInFlash>
</SlotConfig>
</SlotConfigs>
</CySMIFConfiguration>

```

5) Run *cysmifcodegen.exe* (see "Quick Start" section for details).

6) The tool creates two files:

cy_smif_memconfig.h

```

/*****
* \file cy_smif_memconfig.h
* \version 1.0
*
* \brief
* Provides declarations of the SMIF-driver memory configuration.
*
* Note: This is an auto generated file. Do not modify it.
*
*****/
* \copyright
* Copyright 2017, Cypress Semiconductor Corporation. All rights reserved.
* You may use this file only in accordance with the license, terms, conditions,
* disclaimers, and limitations in the end user license agreement accompanying
* the software package with which this file was provided.
*****/
#ifndef CY_SMIF_MEMCONFIG_H
#define CY_SMIF_MEMCONFIG_H
#include "smif/cy_smif_memslot.h"

#define CY_SMIF_DEVICE_NUM 1

extern cy_stc_smif_mem_cmd_t MX25L12835F_0_readCmd;
extern cy_stc_smif_mem_cmd_t MX25L12835F_0_writeEnCmd;
extern cy_stc_smif_mem_cmd_t MX25L12835F_0_writeDisCmd;

```

```
extern cy_stc_smif_mem_cmd_t MX25L12835F_0_eraseCmd;
extern cy_stc_smif_mem_cmd_t MX25L12835F_0_chipEraseCmd;
extern cy_stc_smif_mem_cmd_t MX25L12835F_0_programCmd;
extern cy_stc_smif_mem_cmd_t MX25L12835F_0_readStsRegQeCmd;
extern cy_stc_smif_mem_cmd_t MX25L12835F_0_readStsRegWipCmd;
extern cy_stc_smif_mem_cmd_t MX25L12835F_0_writeStsRegQeCmd;

extern cy_stc_smif_mem_device_cfg_t MX25L12835F_0_DeviceCfg;

extern const cy_stc_smif_mem_config_t MX25L12835F_0;

extern const cy_stc_smif_mem_config_t* smifMemConfigs[CY_SMIF_DEVICE_NUM];

extern const cy_stc_smif_block_config_t smifBlockConfig;

#endif /*CY_SMIF_MEMCONFIG_H*/

cy_smif_memconfig.c:
/*****
* \file cy_smif_memconfig.c
* \version 1.0
*
* \brief
* Provides definitions of the SMIF-driver memory configuration.
*
* Note: This is an auto generated file. Do not modify it.
*
*****/
* \copyright
* Copyright 2017, Cypress Semiconductor Corporation. All rights reserved.
* You may use this file only in accordance with the license, terms, conditions,
* disclaimers, and limitations in the end user license agreement accompanying
* the software package with which this file was provided.
*****/

#include "cy_smif_memconfig.h"

cy_stc_smif_mem_cmd_t MX25L12835F_0_readCmd =
{
    /* The 8-bit command. 1 x I/O read command. */
    .command = 0xEBU,
    /* The width of the command transfer. */
    .cmdWidth = CY_SMIF_WIDTH_SINGLE,
    /* The width of the address transfer. */
    .addrWidth = CY_SMIF_WIDTH_QUAD,
    /* The 8-bit mode byte. This value is 0xFFFFFFFF when there is no mode present. */
    .mode = 0xFFFFFFFFU,
    /* The width of the mode command transfer. */
    .modeWidth = CY_SMIF_WIDTH_QUAD,
    /* The number of dummy cycles. A zero value suggests no dummy cycles. */
    .dummyCycles = 6U,
    /* The width of the data transfer. */
    .dataWidth = CY_SMIF_WIDTH_QUAD
};

cy_stc_smif_mem_cmd_t MX25L12835F_0_writeEnCmd =
{
    /* The 8-bit command. 1 x I/O read command. */
    .command = 0x06U,
    /* The width of the command transfer. */
    .cmdWidth = CY_SMIF_WIDTH_SINGLE,
```

```

/* The width of the address transfer. */
.addrWidth = CY_SMIF_WIDTH_SINGLE,
/* The 8-bit mode byte. This value is 0xFFFFFFFF when there is no mode present. */
.mode = 0xFFFFFFFFFU,
/* The width of the mode command transfer. */
.modeWidth = CY_SMIF_WIDTH_SINGLE,
/* The number of dummy cycles. A zero value suggests no dummy cycles. */
.dummyCycles = 0U,
/* The width of the data transfer. */
.dataWidth = CY_SMIF_WIDTH_SINGLE
};

cy_stc_smif_mem_cmd_t MX25L12835F_0_writeDisCmd =
{
    /* The 8-bit command. 1 x I/O read command. */
    .command = 0x04U,
    /* The width of the command transfer. */
    .cmdWidth = CY_SMIF_WIDTH_SINGLE,
    /* The width of the address transfer. */
    .addrWidth = CY_SMIF_WIDTH_SINGLE,
    /* The 8-bit mode byte. This value is 0xFFFFFFFF when there is no mode present. */
    .mode = 0xFFFFFFFFFU,
    /* The width of the mode command transfer. */
    .modeWidth = CY_SMIF_WIDTH_SINGLE,
    /* The number of dummy cycles. A zero value suggests no dummy cycles. */
    .dummyCycles = 0U,
    /* The width of the data transfer. */
    .dataWidth = CY_SMIF_WIDTH_SINGLE
};

cy_stc_smif_mem_cmd_t MX25L12835F_0_eraseCmd =
{
    /* The 8-bit command. 1 x I/O read command. */
    .command = 0x20U,
    /* The width of the command transfer. */
    .cmdWidth = CY_SMIF_WIDTH_SINGLE,
    /* The width of the address transfer. */
    .addrWidth = CY_SMIF_WIDTH_SINGLE,
    /* The 8-bit mode byte. This value is 0xFFFFFFFF when there is no mode present. */
    .mode = 0xFFFFFFFFFU,
    /* The width of the mode command transfer. */
    .modeWidth = CY_SMIF_WIDTH_SINGLE,
    /* The number of dummy cycles. A zero value suggests no dummy cycles. */
    .dummyCycles = 0U,
    /* The width of the data transfer. */
    .dataWidth = CY_SMIF_WIDTH_SINGLE
};

cy_stc_smif_mem_cmd_t MX25L12835F_0_chipEraseCmd =
{
    /* The 8-bit command. 1 x I/O read command. */
    .command = 0x60U,
    /* The width of the command transfer. */
    .cmdWidth = CY_SMIF_WIDTH_SINGLE,
    /* The width of the address transfer. */
    .addrWidth = CY_SMIF_WIDTH_SINGLE,
    /* The 8-bit mode byte. This value is 0xFFFFFFFF when there is no mode present. */
    .mode = 0xFFFFFFFFFU,
    /* The width of the mode command transfer. */
    .modeWidth = CY_SMIF_WIDTH_SINGLE,
    /* The number of dummy cycles. A zero value suggests no dummy cycles. */
    .dummyCycles = 0U,
    /* The width of the data transfer. */
    .dataWidth = CY_SMIF_WIDTH_SINGLE
};

```

```

        .dataWidth = CY_SMIF_WIDTH_SINGLE
    };

cy_stc_smif_mem_cmd_t MX25L12835F_0_programCmd =
{
    /* The 8-bit command. 1 x I/O read command. */
    .command = 0x38U,
    /* The width of the command transfer. */
    .cmdWidth = CY_SMIF_WIDTH_SINGLE,
    /* The width of the address transfer. */
    .addrWidth = CY_SMIF_WIDTH_QUAD,
    /* The 8-bit mode byte. This value is 0xFFFFFFFF when there is no mode present. */
    .mode = 0xFFFFFFFFU,
    /* The width of the mode command transfer. */
    .modeWidth = CY_SMIF_WIDTH_SINGLE,
    /* The number of dummy cycles. A zero value suggests no dummy cycles. */
    .dummyCycles = 0U,
    /* The width of the data transfer. */
    .dataWidth = CY_SMIF_WIDTH_QUAD
};

cy_stc_smif_mem_cmd_t MX25L12835F_0_readStsRegQeCmd =
{
    /* The 8-bit command. 1 x I/O read command. */
    .command = 0x05U,
    /* The width of the command transfer. */
    .cmdWidth = CY_SMIF_WIDTH_SINGLE,
    /* The width of the address transfer. */
    .addrWidth = CY_SMIF_WIDTH_SINGLE,
    /* The 8-bit mode byte. This value is 0xFFFFFFFF when there is no mode present. */
    .mode = 0xFFFFFFFFU,
    /* The width of the mode command transfer. */
    .modeWidth = CY_SMIF_WIDTH_SINGLE,
    /* The number of dummy cycles. A zero value suggests no dummy cycles. */
    .dummyCycles = 0U,
    /* The width of the data transfer. */
    .dataWidth = CY_SMIF_WIDTH_SINGLE
};

cy_stc_smif_mem_cmd_t MX25L12835F_0_readStsRegWipCmd =
{
    /* The 8-bit command. 1 x I/O read command. */
    .command = 0x05U,
    /* The width of the command transfer. */
    .cmdWidth = CY_SMIF_WIDTH_SINGLE,
    /* The width of the address transfer. */
    .addrWidth = CY_SMIF_WIDTH_SINGLE,
    /* The 8-bit mode byte. This value is 0xFFFFFFFF when there is no mode present. */
    .mode = 0xFFFFFFFFU,
    /* The width of the mode command transfer. */
    .modeWidth = CY_SMIF_WIDTH_SINGLE,
    /* The number of dummy cycles. A zero value suggests no dummy cycles. */
    .dummyCycles = 0U,
    /* The width of the data transfer. */
    .dataWidth = CY_SMIF_WIDTH_SINGLE
};

cy_stc_smif_mem_cmd_t MX25L12835F_0_writeStsRegQeCmd =
{
    /* The 8-bit command. 1 x I/O read command. */
    .command = 0x01U,
    /* The width of the command transfer. */
    .cmdWidth = CY_SMIF_WIDTH_SINGLE,

```

```

/* The width of the address transfer. */
.addrWidth = CY_SMIF_WIDTH_SINGLE,
/* The 8-bit mode byte. This value is 0xFFFFFFFF when there is no mode present. */
.mode = 0xFFFFFFFFU,
/* The width of the mode command transfer. */
.modeWidth = CY_SMIF_WIDTH_SINGLE,
/* The number of dummy cycles. A zero value suggests no dummy cycles. */
.dummyCycles = 0U,
/* The width of the data transfer. */
.dataWidth = CY_SMIF_WIDTH_SINGLE
};

cy_stc_smif_mem_device_cfg_t deviceCfg_MX25L12835F_0 =
{
    /* Specifies the number of address bytes used by the memory slave device. */
    .numOfAddrBytes = 0x03U,
    /* The size of the memory. */
    .memSize = 0x1000000U,
    /* Specifies the Read command. */
    .readCmd = &MX25L12835F_0_readCmd,
    /* Specifies the Write Enable command. */
    .writeEnCmd = &MX25L12835F_0_writeEnCmd,
    /* Specifies the Write Disable command. */
    .writeDisCmd = &MX25L12835F_0_writeDisCmd,
    /* Specifies the Erase command. */
    .eraseCmd = &MX25L12835F_0_eraseCmd,
    /* Specifies the sector size of each erase. */
    .eraseSize = 0x0001000U,
    /* Specifies the Chip Erase command. */
    .chipEraseCmd = &MX25L12835F_0_chipEraseCmd,
    /* Specifies the Program command. */
    .programCmd = &MX25L12835F_0_programCmd,
    /* Specifies the page size for programming. */
    .programSize = 0x0000100U,
    /* Specifies the command to read the QE-containing status register. */
    .readStsRegQeCmd = &MX25L12835F_0_readStsRegQeCmd,
    /* Specifies the command to read the WIP-containing status register. */
    .readStsRegWipCmd = &MX25L12835F_0_readStsRegWipCmd,
    /* Specifies the command to write into the QE-containing status register. */
    .writeStsRegQeCmd = &MX25L12835F_0_writeStsRegQeCmd,
    /* The mask for the status register. */
    .stsRegBusyMask = 0x01U,
    /* The mask for the status register. */
    .stsRegQuadEnableMask = 0x40U,
    /* The max time for the erase type-1 cycle-time in ms. */
    .eraseTime = 120U,
    /* The max time for the chip-erase cycle-time in ms. */
    .chipEraseTime = 80000U,
    /* The max time for the page-program cycle-time in us. */
    .programTime = 1500U
};

const cy_stc_smif_mem_config_t MX25L12835F_SlaveSlot_0 =
{
    /* Determines the slot number where the memory device is placed. */
    .slaveSelect = CY_SMIF_SLAVE_SELECT_0,
    /* Flags. */
    .flags = CY_SMIF_FLAG_MEMORY_MAPPED,
    /* The data-line selection options for a slave device. */
    .dataSelect = CY_SMIF_DATA_SELO,
    /* The base address the memory slave is mapped to in the PSoC memory map.
    Valid when the memory-mapped mode is enabled. */
    .baseAddress = 0x18000000U,

```



```
/* The size allocated in the PSoC memory map, for the memory slave device.
The size is allocated from the base address. Valid when the memory mapped mode is
enabled. */
.memMappedSize = 0x10000U,
/* If this memory device is one of the devices in the dual quad SPI configuration.
Valid when the memory mapped mode is enabled. */
.dualQuadSlots = 0,
/* The configuration of the device. */
.deviceCfg = &deviceCfg_MX25L12835F_0
};

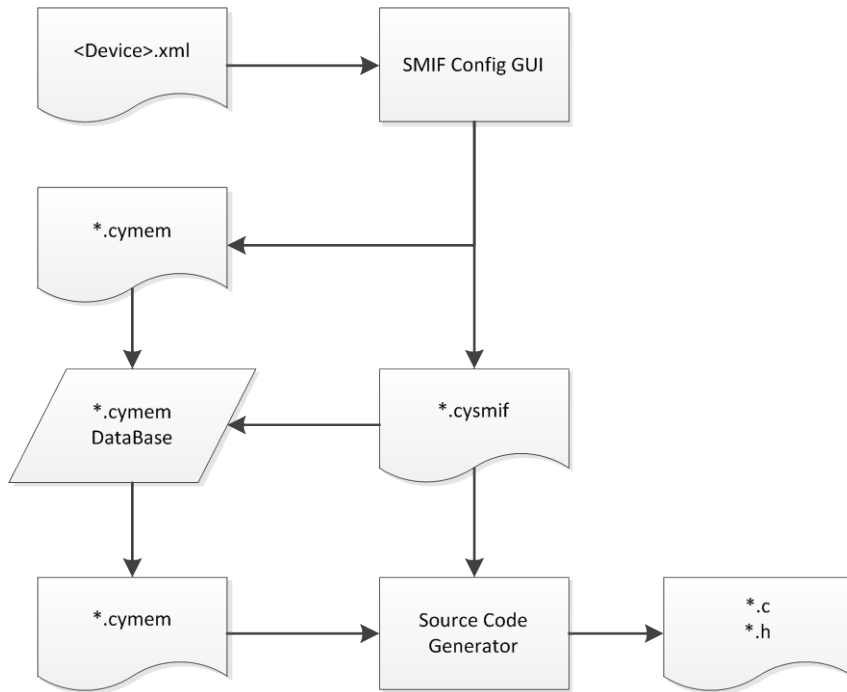
const cy_stc_smif_mem_config_t* smifMemConfigs[] = {
    &MX25L12835F_SlaveSlot_0
};

const cy_stc_smif_block_config_t smifBlockConfig =
{
    /* The number of SMIF memories defined. */
    .memCount = CY_SMIF_DEVICE_NUM,
    /* The pointer to the array of memory config structures of size memCount. */
    .memConfig = (cy_stc_smif_mem_config_t**)smifMemConfigs,
    /* The version of the SMIF driver. */
    .majorVersion = CY_SMIF_DRV_VERSION_MAJOR,
    /* The version of the SMIF driver. */
    .minorVersion = CY_SMIF_DRV_VERSION_MINOR
};
```

4. XML Description



The SMIF configuration tool works with multiple XML files. This section provides an overview of the structure of these XML files.



4.1 *.cysmif File

The *.cysmif file contains the configuration data for the entire SMIF block. This XML file provides all the PSoC-specific information such as the start address and size. For each memory device that is connected to the different slave slots of the SMIF, this XML file provides a link to the corresponding memory's *.cymem file.

Schema:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="CySMIFConfiguration">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="DevicePath"/>
        <xs:element name="SlotConfigs">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="SlotConfig" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element type="xs:byte" name="SlaveSlot"/>
                    <xs:element type="xs:string" name="PartNumber"/>
                    <xs:element type="xs:string" name="MemoryMapped"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```

        <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:string" name="WriteEnable" />
        <xs:element type="xs:string" name="Encrypt" />
        <xs:element type="xs:string" name="DataSelect" />
        <xs:element type="xs:string" name="Description" />
        <xs:element type="xs:string" name="MemoryConfigsPath" />
        <xs:element type="xs:string" name="ConfigDataInFlash" />
    </xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

4.2 *.cymem File

This XML file provides the configuration data that is specific to the memory device. Configuration parameters in this XML file are mostly derived from the memory device's datasheet. There will be a *.cymem file associated with every memory device supported by the SMIF configuration tool.

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="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">
                    <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:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>

```

```

        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element type="xs:string" name="IsSupportedMemory"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema></xs:schema>

```

4.3 GUI, XML Parameters, and C-Structures

This section describes the connection between the data input in the GUI and data in the generated CYMEM/CYSMIF files, as well as in the generated SMIF driver structures. For a description of the SMIF driver structures, fields, and field values, refer to the SMIF driver section in the PDL API Reference Guide.

4.3.1 Slot Parameters (*.cysmif file)

Parameter	Description	C-Structure (<code>cy_stc_smif_mem_config_t</code>)
GUI parameter: Slave slot XML tag: <SlaveSlot> Type: int	Specifies the slave slot being configured. This number corresponds to the slave select line that will be connected to the memory device.	slaveSelect set of [0, 1, 2, 3] as [1, 2, 4, 8]
GUI parameter: Memory part number XML tag: <PartNumber> - file name <MemoryConfigsPath> - file path Type: enum (memory file names)	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	Used for structure naming
GUI parameter: Data select XML tag: <DataSelect> Type: enum (CY_SMIF_DATA_SEL)	Lets you select the data line options for the corresponding memory slot.	dataSelect
GUI parameter: Memory mapped XML tag: <MemoryMapped> Type: bool	When this option is enabled, the configured memory device is mapped to the PSoC device's memory map. If disabled, access to memory must be done through the SMIF API.	flags
GUI parameter: Pair with slot XML tag: <DualQuad> Type: enum (int or "NONE") - slot number	Determines the paired slot for dual Quad operation	dualQuadSlots CY_SMIF_SLAVE_SELECT_X CY_SMIF_SLAVE_SELECT_Y X - 1st memory in pair Y - 2nd memory in pair
GUI parameter: Start address XML tag: <StartAddress> Type: string ("0x...") - hex value	Determines the starting address where the memory device is going to be mapped in the PSoC memory map.	baseAddress = StartAddress memMappedSize = EndAddress - StartAddress
GUI parameter: Size XML tag: <Size> Type: string ("0x...") - hex value	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.	baseAddress = StartAddress memMappedSize = EndAddress - StartAddress
GUI parameter: End address XML tag: <EndAddress> Type: string ("0x...") - hex value	Represents the end address of the memory device as mapped in the PSoC device's memory map. This field is a representation of the size of the memory being mapped.	baseAddress = StartAddress memMappedSize = EndAddress - StartAddress
GUI parameter: Write enable XML tag: <WriteEnable> Type: bool	Lets you enable or disable writes to the external memory in a memory mapped mode of operation.	flags
GUI parameter: Config Data in Flash t XML tag: < ConfigDataInFlash > Type: bool	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.	Initialize structure as constant

Parameter	Description	C-Structure (cy_stc_smif_mem_config_t)
GUI parameter: Encrypt XML tag: <Encrypt> Type: bool	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.	flags

4.3.2 Memory Parameters (*.cymem file)

Parameter	Description	C-Structure (cy_stc_smif_mem_device_cfg_t)
GUI parameter: User part number XML tag: <PartNumber> Type: string	Device part number represents the memory device part number.	Is used for structure naming.
GUI parameter: Status register busy mask XML tag: <StatusRegisterBusyMask> Type: range (0x00 - 0xFF)	Mask for the busy bit in the status register.	stsRegBusyMask
GUI parameter: Status register quad enable mask XML tag: <StatusRegisterQuadEnableMask> Type: range (0x00 - 0xFF)	Mask for the quad enable bit in the status register.	stsRegQuadEnableMask
GUI parameter: Size of memory XML tag: <SizeOfMemory> Type: range (256 bytes - 256M bytes)	Denotes the actual size of the memory device.	memSize
GUI parameter: Program page size XML tag: <ProgramPageSize> Type: range (1 byte - 256 Mbytes)	Denotes the page size for a program operation. This size provides the granularity with which program operations can be committed in the memory device.	programSize
GUI parameter: Erase block size XML tag: <EraseBlockSize> Type: range (1 byte - 256 Mbytes)	Provides the erase block size.	eraseSize
GUI parameter: Number of address bytes for SMIF transactions XML tag: <NumberOfAddress> Type: range (1 - 4 bytes)	Sets the number of bytes that are expected for the address field in the SMIF transaction.	numOfAddrBytes
GUI parameter: Erase time XML tag: <EraseTime> Type: range (1ms - 32s)	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.	eraseTime
GUI parameter: Chip erase time XML tag: <ChipEraseTime> Type: range (16ms - 2048s)	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.	chipEraseTime
GUI parameter: Program time XML tag: <ProgramTime> Type: range (8us - 2048us)	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.	programTime

4.3.3 Command Parameters (*.cymem file)

Parameter	Description	C-Structure (cy_stc_smif_mem_cmd_t)
GUI parameter: Number XML tag: <CommandNumber> Type: range (0x00 - 0xFF)	Byte command word.	Command
GUI parameter: Command width XML tag: <CmdWidth> Type: enum: (CY_SMIF_WIDTH)	Width of the command transfer	cmdWidth
GUI parameter: Address width XML tag: <AddrWidth> Type: enum: (CY_SMIF_WIDTH)	Width of the address transfer	addrWidth
GUI parameter: Mode XML tag: <Mode> Type: range (0x00 - 0xFF)	Provides the mode word for the command	Mode In the *.cymem file and C-structure value is set 0xFFFFFFFF if not available
GUI parameter: Mode width XML tag: <ModeWidth> Type: enum (CY_SMIF_WIDTH)	Provides the width of the mode word transfer	modeWidth
GUI parameter: Dummy cycles XML tag: <DummyCycles> Type: range 0 - 32 cycles (In *.cymem file and C-structure value is set 0 if not available)	Provides the number of dummy cycles in the transfer	dummyCycles
GUI parameter: Data width XML tag: <DataWidth> Type: enum (CY_SMIF_WIDTH)	Provides the width of data bytes in the transfer	dataWidth

4.3.4 Command List (*.cymem file)

Parameter	Description	C-Structure (cy_stc_smif_mem_device_cfg_t)
ReadCmd	Read command format	readCmd
WriteEnCmd	Write enable command format	writeEnCmd
WriteDisCmd	Write disable command format	writeDisCmd
SectEraseCmd	Erase command format	eraseCmd
ChipEraseCmd	Chip erase command format	chipEraseCmd
ProgramCmd	Program command format	programCmd
readStsRegQeCmd	Read status register command (containing QE bit)	readStsRegQeCmd
readStsRegWipCmd	Read status register command (containing WIP bit)	readStsRegWipCmd
writeStsRegQeCmd	Write status register command (containing QE bit)	writeStsRegQeCmd

5. Error Handling



Code	Message	Solution
0000	File <file name> is incorrect.	XML file cannot be deserialized. Please check: <ul style="list-style-type: none"> file name file location XML structure is not corrupted.
0001	XML validation error	XML file structure mismatched with schema (See section 4. XML Description). Please fix XML structure or generate new file.
0010	The slot is "Not used".	Choose "Memory part number" for this slot.
0020	The value is not in the HEX format or too big.	Parameter value should be in format: 0x00000000.
0021	The value of the <i>parameter name</i> parameter should be in the HEX format.	
0022	The value is not integer or too large.	Fix the value format. Ensure that there are no characters, comma, or dots.
0023	The value of the <i>parameter name</i> parameter should be integer.	
0030	The end address should be a multiple of 2.	Base address of the SMIF XIP memory region. This address must be a multiple of the SMIF XIP memory capacity. This address must be a multiple of the SMIF XIP memory region capacity (see SMIF_XIP_MASK below). The SMIF XIP memory region should NOT overlap with other memory regions. This address must be in the [0x0000:0000, 0xffff:0000] memory region. However, for MXS40, this address must be in the [0x0000:0000, 0x1fff:0000] memory region (to ensure a connection to the ARM CM4 CPU ICode/DCode memory region [0x0000:0000, 0x1fff:ffff]). The external memory devices are located within the SMIF XIP memory region. Capacity of the SMIF XIP memory region. The capacity must be a power of 2 and greater or equal than 64 KB).
0031	The slot memory size must be 64KB or more.	
0032	The start address should be a multiple of the slot memory size.	
0033	The start address must be less the end address.	
0034	The value of the <i>Start/End Address/Size</i> parameter should be more than <i>min value</i> and less than <i>max value</i> .	

Revision History



Document Title: SMIF Configuration Tool 1.0 User Guide Document Number: 002-19118			
Revision	Issue Date	Origin of Change	Description of Change
**	04/04/2017	ASPT	Initial release
*A	06/30/2017	ASPT	Updated GUI screenshots. Added Linux and MAC cysmifcodegen usage. Added example of full generating flow. Updated XML schemas. Added "Size" and "Config Data in Flash" parameter description. Fixed "Program page size", "Erase block size", and "Dummy cycles" range. Fixed Commands description. Added list of errors with codes and solutions.
*B	10/09/2017	ASPT	Added "Memory Database" section