User Manual

for S32K1 FLS Driver

Document Number: UM2FLSASR4.4 Rev0000R1.0.1 Rev. 1.0

Chapter 1

Revision History

Revision	Date	Author	Description
1.0	24.02.2022	NXP RTD Team	Prepared for release RTD S32K1 Version 1.0.1

Chapter 2

Introduction

This User Manual describes NXP Semiconductor AUTOSAR FLS for S32K1. AUTOSAR FLS driver configuration parameters and deviations from the specification are described in Driver chapter of this document. AUTOSAR FLS driver requirements and APIs are described in the AUTOSAR FLS driver software specification document.

2.1 Supported Derivatives

The software described in this document is intended to be used with the following microcontroller devices of NXP Semiconductors:

- s32k116_qfn32
- s32k116_lqfp48
- s32k118_lqfp48
- s32k118_lqfp64
- s32k142_lqfp48
- s32k142_lqfp64
- s32k142_lqfp100
- s32k142w_lqfp48
- s32k142w_lqfp64
- s32k144_lqfp48

- s32k144_lqfp64
- s32k144_lqfp100
- s32k144_mapbga100
- s32k144w_lqfp48
- s32k144w_lqfp64
- s32k146_lqfp64
- s32k146_lqfp100
- s32k146_mapbga100
- s32k146_lqfp144
- s32k148_lqfp100
- s32k148_mapbga100
- s32k148_lqfp144
- s32k148_lqfp176

All of the above microcontroller devices are collectively named as S32K1.

2.2 Overview

AUTOSAR (AUTomotive Open System ARchitecture) is an industry partnership working to establish standards for software interfaces and software modules for automobile electronic control systems.

AUTOSAR:

- paves the way for innovative electronic systems that further improve performance, safety and environmental friendliness.
- is a strong global partnership that creates one common standard: "Cooperate on standards, compete on implementation".
- is a key enabling technology to manage the growing electrics/electronics complexity. It aims to be prepared for the upcoming technologies and to improve cost-efficiency without making any compromise with respect to quality.
- facilitates the exchange and update of software and hardware over the service life of the vehicle.

2.3 About This Manual

This Technical Reference employs the following typographical conventions:

- **Boldface** style: Used for important terms, notes and warnings.
- *Italic* style: Used for code snippets in the text. Note that C language modifiers such "const" or "volatile" are sometimes omitted to improve readability of the presented code.

Notes and warnings are shown as below:

Note

This is a note.

Warning

This is a warning

2.4 Acronyms and Definitions

Term	Definition
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
DET	Default Error Tracer
ECC	Error Correcting Code
VLE	Variable Length Encoding
N/A	Not Available
MCU	Microcontroller Unit
ECU	Electronic Control Unit
EEPROM	Electrically Erasable Programmable Read-Only Memory
FEE	Flash EEPROM Emulation
FLS	Flash
RTD	Real Time Drivers
XML	Extensible Markup Language

2.5 Reference List

#	Title	Version
1	Specification of Fls Driver	AUTOSAR Release 4.4.0
2	Reference Manual	S32K1xx Series Reference Manual, Rev. 14, 09/2021
3	Datasheet	S32K1xx Data Sheet, Rev. 14, 08/2021
4	Errata	S32K116_0N96V Rev. 22/OCT/2021
		S32K118_0N97V Rev. 22/OCT/2021
		S32K142_0N33V Rev. 22/OCT/2021
		S32K144_0N57U Rev. 22/OCT/2021
		S32K144W_0P64A Rev. 22/OCT/2021
		S32K146_0N73V Rev. 22/OCT/2021
		S32K148_0N20V Rev. 22/OCT/2021

Chapter 3

Driver

3.1 Requirements

Requirements for this driver are detailed in the Autosar Driver Software Specification document (See Table Reference List).

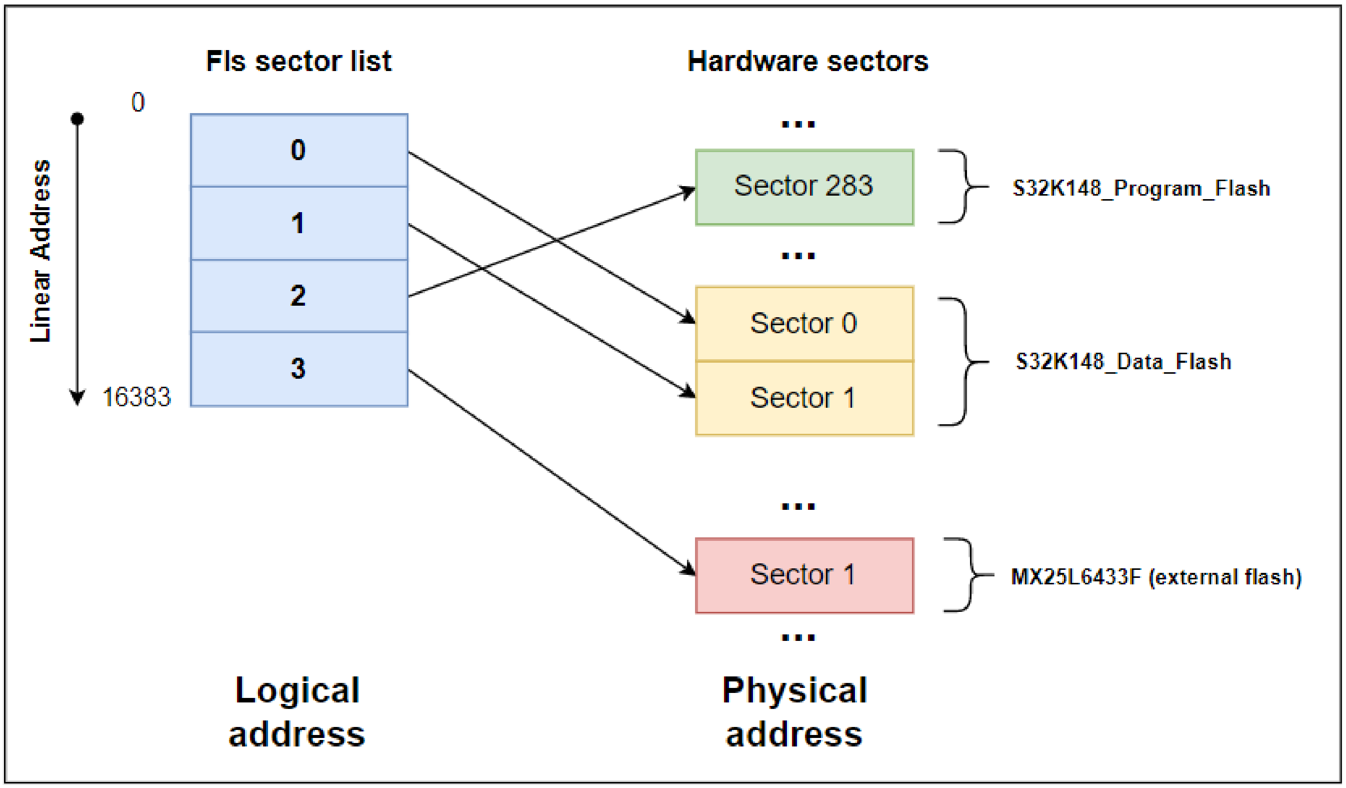
3.2 Driver Design Summary

3.2.1 Linear Address

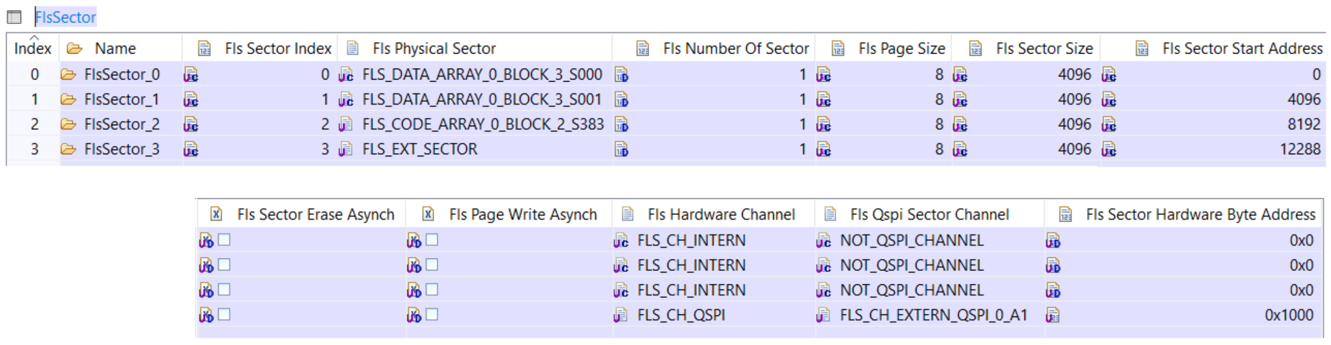
- The FLS driver provides services for reading, writing and erasing flash memory and it combines configured flash memory sectors into one linear address space. The FLS module shall combine all available flash memory areas into one linear address space, it will always start at address 0 and continues without any gap.
- Sectors details Example, suppose user wants to configure following sectors:

FlsPhysicalSector	Fls Physical Start Address	Fls Sector Size
FlsSector_0	0 (0x0000)	4096 (0x1000)
FlsSector_1	4096 (0x1000)	4096 (0x1000)
FlsSector_2	8192 (0x2000)	4096 (0x1000)
FlsSector_3	12288 (0x3000)	4096 (0x1000)

The layout of configured sectors:



The FlsSector List should be configured in the following way:


FlsSector

Index	Name	Fls Sector Index	Fls Physical Sector	Fls Number Of Sector	Fls Page Size	Fls Sector Size	Fls Sector Start Address
0	FlsSector_0	0	FLS_DATA_ARRAY_0_BLOCK_3_S000	1	8	4096	0
1	FlsSector_1	1	FLS_DATA_ARRAY_0_BLOCK_3_S001	1	8	4096	4096
2	FlsSector_2	2	FLS_CODE_ARRAY_0_BLOCK_2_S383	1	8	4096	8192
3	FlsSector_3	3	FLS_EXT_SECTOR	1	8	4096	12288

Fls Sector Erase Asynch	Fls Page Write Asynch	Fls Hardware Channel	Fls Qspi Sector Channel	Fls Sector Hardware Byte Address
☐	☐	FLS_CH_INTERN	NOT_QSPI_CHANNEL	0x0
☐	☐	FLS_CH_INTERN	NOT_QSPI_CHANNEL	0x0
☐	☐	FLS_CH_INTERN	NOT_QSPI_CHANNEL	0x0
☐	☐	FLS_CH_QSPI	FLS_CH_EXTERN_QSPI_0_A1	0x1000

- As you can see Fls Sector Start Address for FlsSector_0 will be 0x0000 and Fls Sector Start Address for FlsSector_1 will be 0x1000 (4096) and so on.
- If user want to write FlsSector_1, user need to write to the logical address 0x1000 - 0x1FFF.
- If user want to erase it, user need to erase sector from address 0x1000 with size 0x1000.

Note

- The users do not need to calculate the Fls Sector Start Address and Fls Sector Size they can be automatically computed.
- If the microcontroller is in user mode, be sure that the Flash memory controller registers are accessible. For more information please refer to the 'Memory Protection Unit' and 'Register Protection' chapters in the device reference manual
- Care should be taken if using the first program flash sector (FLS_CODE_ARRAY_0_BLOCK_0_S000), as that sector contains the flash configuration field.
  - It is recommended to avoid using this sector, because the flash configuration field region has a default value different than the default erased value.
  - Writing in improper value into the flash configuration field region (0x0_0400 - 0x0_040F) might lead to protected sectors, secured device or permanently secured device.
  - An example of unprotecting the internal flash sectors is provided in the 'Driver usage and configuration tips' chapter.

3.2.2 Fls Programming Size

- Every write access to the underlying hardware memory will be done by using writes that are as big as allowed on the hardware
  - Internal flash: program phrase command of Flash Memory Module FTFx (8 bytes)
  - External flash: maximum of QSPI Tx buffer size (128 bytes)
  - Fls QuadSPI driver architecture

3.2.3 Fls QuadSPI driver architecture

This section describes the detail for a high-level overview of QuadSPI components in Fls driver, how they interact and how the driver should be used.

Table of content:

- High-level overview
- Use cases
- Supported memories

Related information:

- Clocking and IOMUX for QuadSPI (chapter "3.3 Hardware Resources" in User Manual)
- QuadSPI in multicore context (if supported by the platform, chapter "5.8 Multicore support" in Integration Manual)
- QuadSPI external memory assumptions (chapter "9 External assumptions for driver" in Integration Manual)

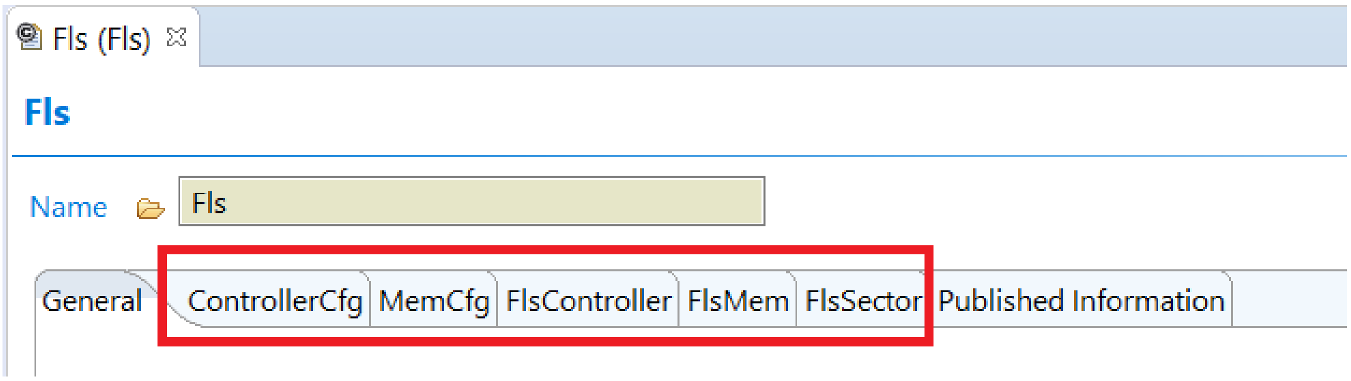
3.2.3.1 High-level overview

This sub-chapter describes the the architecture of the driver:

- The interaction between the HLD, Controller and Memory components
- What each part does and how they interact
- Examples of configuration

3.2.3.1.1 QuadSPI components in Fls driver

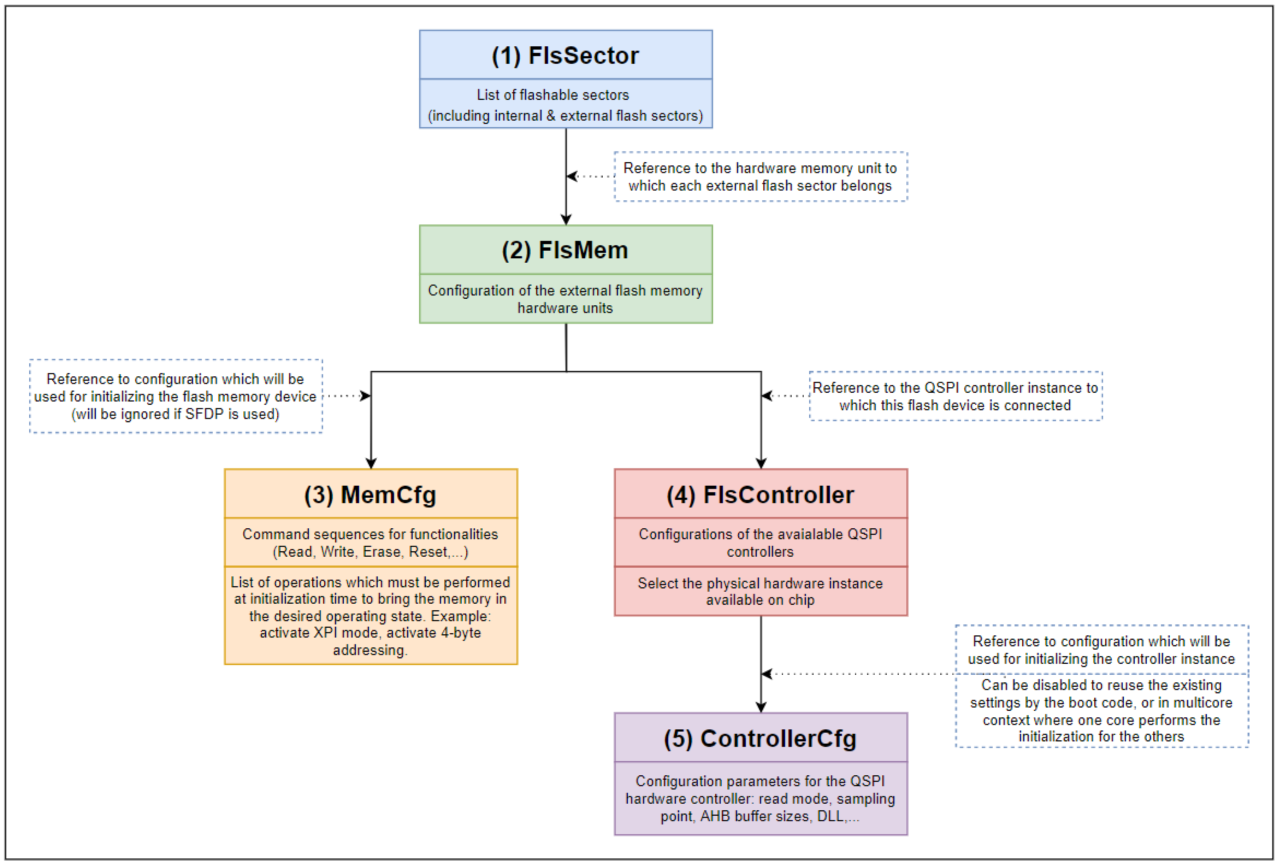
QuadSPI components user interface:



There are five components connect together in the order below:

#	Device name
1	FlsSector
2	FlsMem
3	MemCfg
4	FlsController
5	ControllerCfg

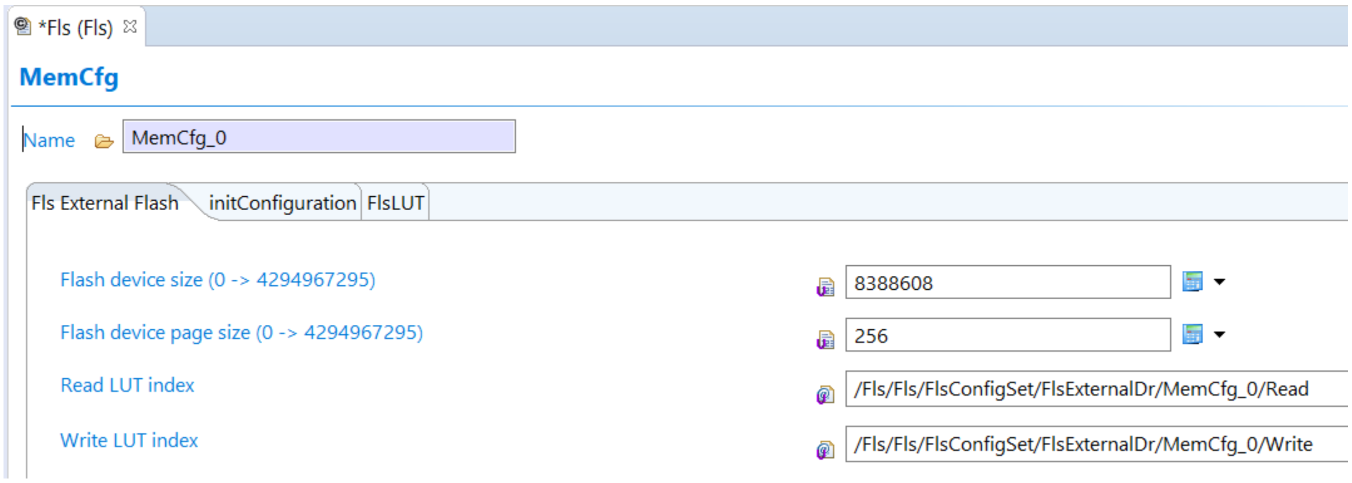
Connections between components:


(1) FlsSector
List of flashable sectors (including internal & external flash sectors)
Reference to the hardware memory unit to which each external flash sector belongs
(2) FlsMem
Configuration of the external flash memory hardware units
Reference to configuration which will be used for initializing the flash memory device (will be ignored if SFDP is used)
Reference to the QSPI controller instance to which this flash device is connected
(3) MemCfg
Command sequences for functionalities (Read, Write, Erase, Reset,...)
List of operations which must be performed at initialization time to bring the memory in the desired operating state. Example: activate XPI mode, activate 4-byte addressing.
(4) FlsController
Configurations of the avaialable QSPI controllers
Select the physical hardware instance available on chip
Reference to configuration which will be used for initializing the controller instance
Can be disabled to reuse the existing settings by the boot code, or in multicore context where one core performs the initialization for the others
(5) ControllerCfg
Configuration parameters for the QSPI hardware controller: read mode, sampling point, AHB buffer sizes, DLL,...

3.2.3.1.2 MemCfg

This container contains all specific settings for the memory device:

- Memory characteristics: device size, page size
- LUT command sequences for basic functionality

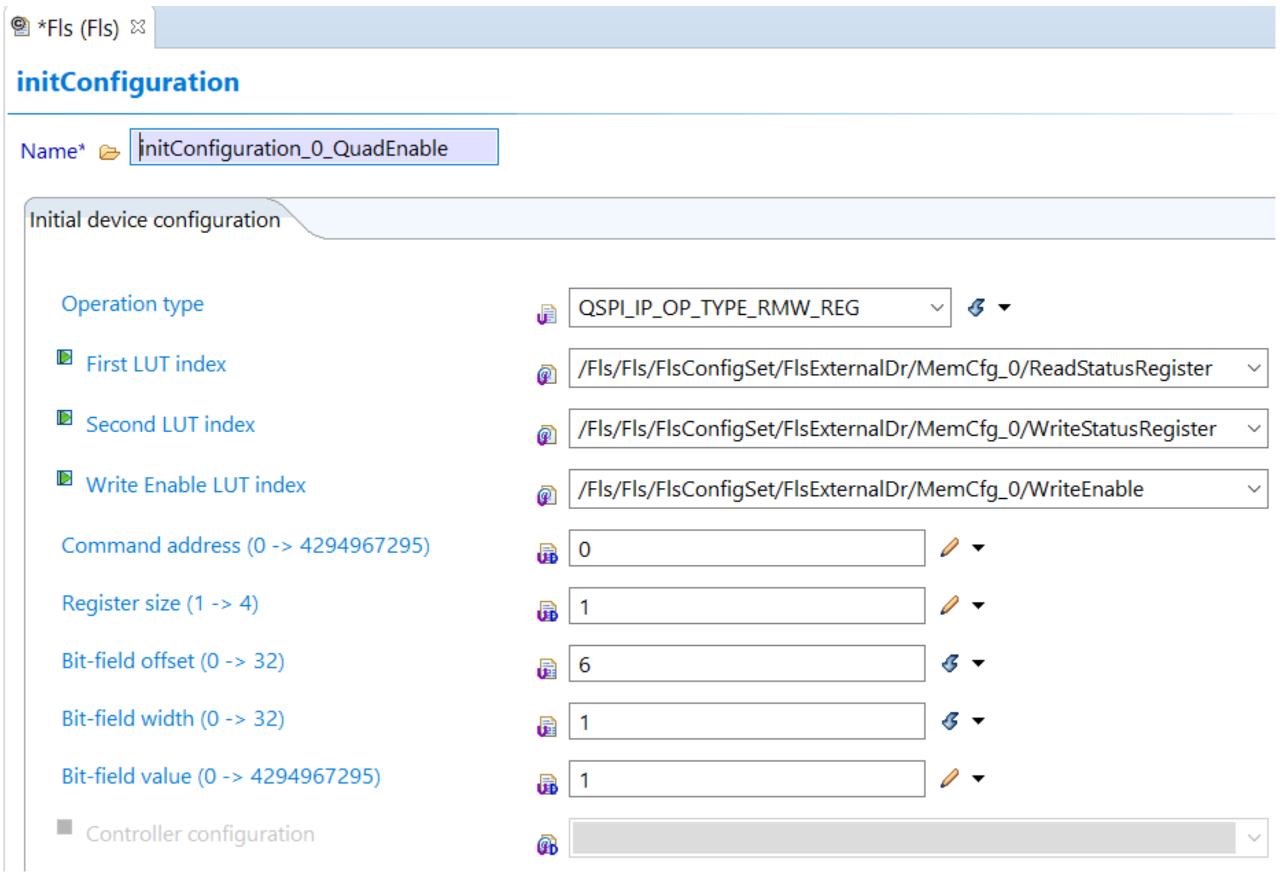


MemCfg

Name MemCfg_0

Flash device size (0 -> 4294967295)	8388608
Flash device page size (0 -> 4294967295)	256
Read LUT index	/Fls/Fls/FlsConfigSet/FlsExternalDr/MemCfg_0/Read
Write LUT index	/Fls/Fls/FlsConfigSet/FlsExternalDr/MemCfg_0/Write

It also provides a list of operations (**initConfiguration**) which must be performed at initialization time to bring the memory in the desired operating state (for example: setting registers value). Here is an example of an operation to enable the Quad mode by set bit 6th in the Status register:


*Fls (Fls)

initConfiguration

Name* initConfiguration_0_QuadEnable

Initial device configuration

Operation type	QSPI_IP_OP_TYPE_RMW_REG
First LUT index	/Fls/Fls/FlsConfigSet/FlsExternalDr/MemCfg_0/ReadStatusRegister
Second LUT index	/Fls/Fls/FlsConfigSet/FlsExternalDr/MemCfg_0/WriteStatusRegister
Write Enable LUT index	/Fls/Fls/FlsConfigSet/FlsExternalDr/MemCfg_0/WriteEnable
Command address (0 -> 4294967295)	0
Register size (1 -> 4)	1
Bit-field offset (0 -> 32)	6
Bit-field width (0 -> 32)	1
Bit-field value (0 -> 4294967295)	1
Controller configuration	

In this example, QuadSPI driver will:

- Read 1 byte value of the status register by using the **First LUT index**
- Modify the **6th** bit to the desired value is **1**
- If needed, the **Write Enable LUT index** will be issued before a write command
- Write back that byte value to memory device by using the **Second LUT index**

Note

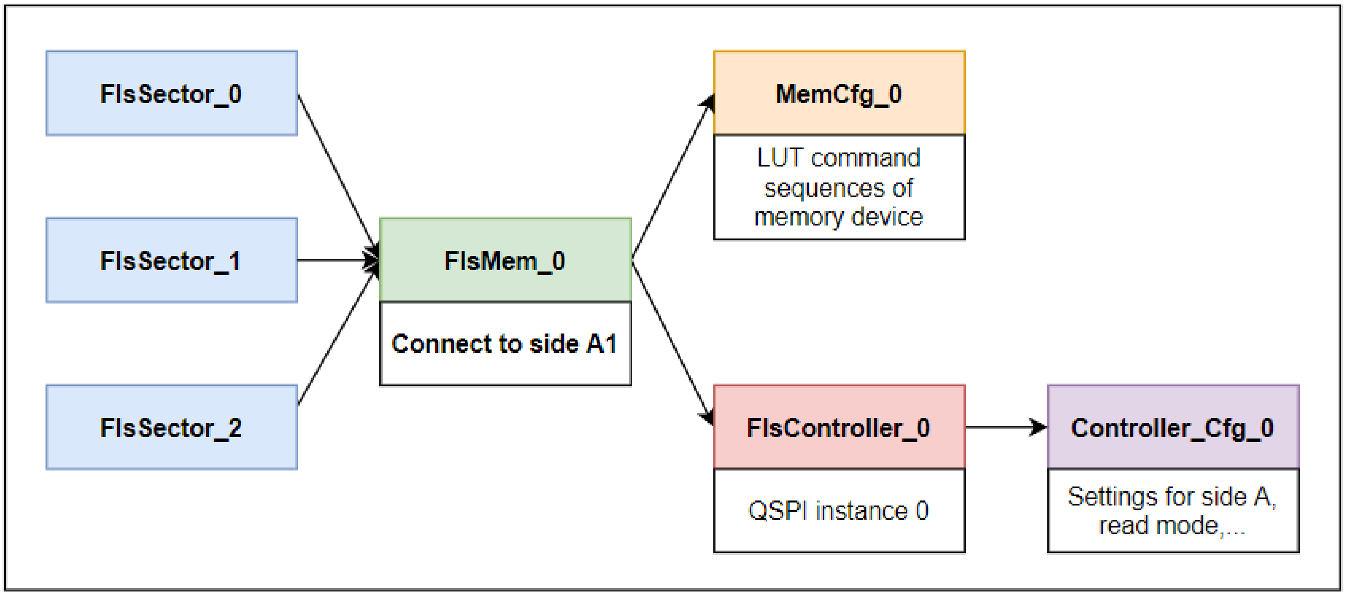
- When changing the value of non-volatile bits, users need to insert one more read operation (QSPI_IP↩ _OP_TYPE_READ_REG) right behind to wait for the write operation to complete

3.2.3.1.3 Examples of configuration

3.2.3.1.3.1 Example 1

Below is the diagram to depict the example from the section "3.2.1 Linear Address". Assume that the memory device connects to the side A1 of QuadSPI controller, we need:

- 01 hardware memory unit (reference to MemCfg_0): contains the LUT command set for initializing the memory device
- 01 controller configuration set (ControllerCfg_0): contains the configuration parameters for QuadSPI hardware instance 0



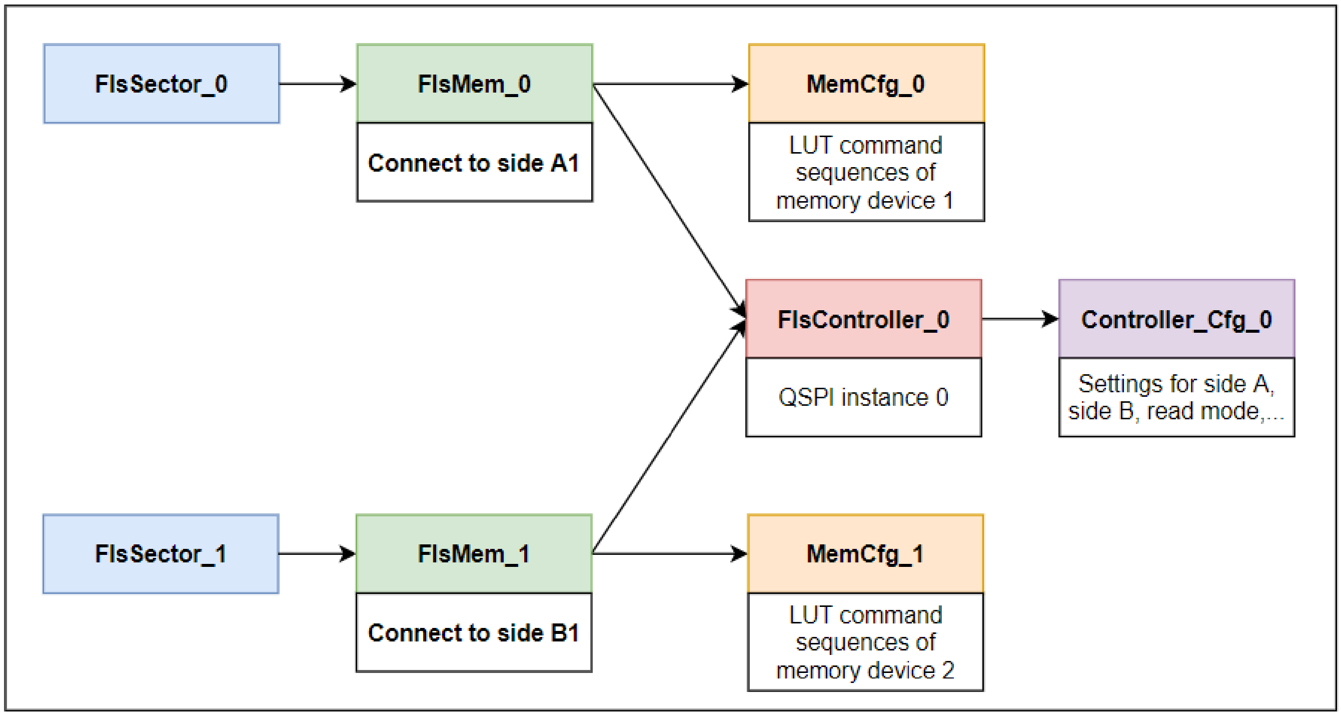
3.2.3.1.3.2 Example 2

Suppose we have 02 different external flash memory devices, one connects to side A and one connects to side B of the QuadSPI controller. And we want to configure 02 external sectors, the first sector in each memory device

FlsSectorList	Fls Sector Size	Fls Number Of Sector	Fls Logical Start Address	Fls Hardware Start Address	Fls Hardware Memory Unit
FlsSector_0	4096 (0x1000)	1	0 (0x0000)	0 (0x0000)	FlsMem_0
FlsSector_1	4096 (0x1000)	1	4096 (0x1000)	0 (0x0000)	FlsMem_1

In this example, we need:

- 02 hardware memory unit (reference to MemCfg_0 and MemCfg_1): contain the LUT command sets for initializing each memory device (if they have different command sets)
- 01 controller configuration set (ControllerCfg_0): contains the configuration parameters for QuadSPI hardware instance 0



As you can see:

- Any operations on the FlsSector_0 will be mapped to the hardware sector 0 (0x0000 - 0x0FFF) of the memory device 1
- Any operations on the FlsSector_1 will be mapped to the hardware sector 0 (0x0000 - 0x0FFF) of the memory device 2
- The QuadSPI controller communicates with each memory device by using the commands set from the corresponding **MemCfg**

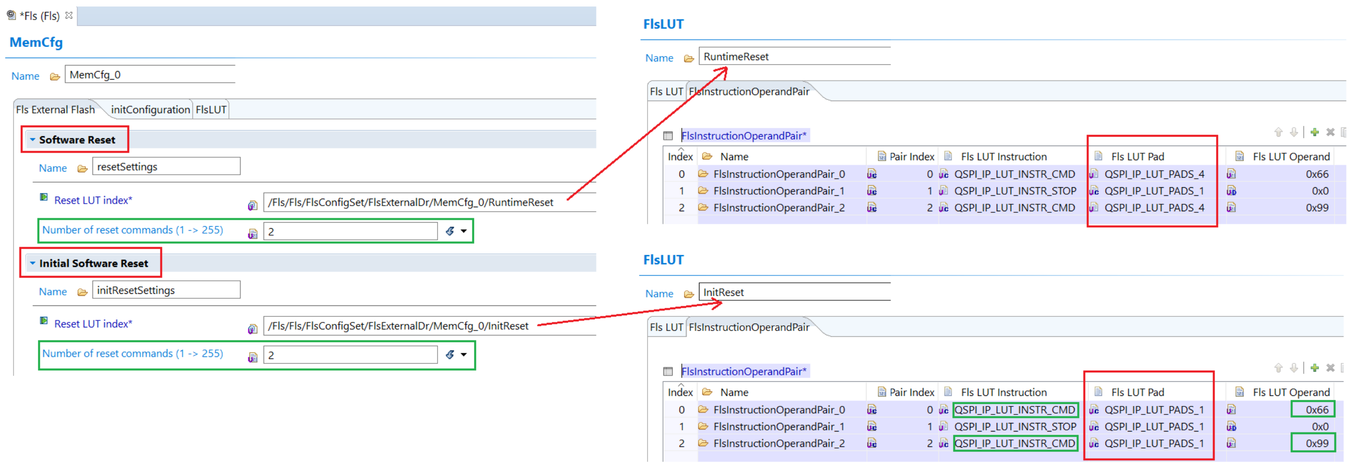
3.2.3.2 Use cases

This sub-chapter provides various useful practical examples.

3.2.3.2.1 Software reset

The driver provides two ways to use the software reset command, for resettings the flash device:

- Software Reset (used by Fls_Cancel() and Qspi_Ip_Reset(), at any time during runtime)
- Initial Software Reset (used by Fls_Init(), only one time)



MemCfg

Name: MemCfg_0

Software Reset

Name: resetSettings

Reset LUT index*: /Fls/Fls/FlsConfigSet/FlsExternalDr/MemCfg_0/RuntimeReset

Number of reset commands (1 -> 255): 2

Initial Software Reset

Name: initResetSettings

Reset LUT index*: /Fls/Fls/FlsConfigSet/FlsExternalDr/MemCfg_0/InitReset

Number of reset commands (1 -> 255): 2

FlsLUT

Name: RuntimeReset

FlsInstructionOperandPair*

Index	Name	Pair Index	Fls LUT Instruction	Fls LUT Pad	Fls LUT Operand
0	FlsInstructionOperandPair_0	0	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_4	0x66
1	FlsInstructionOperandPair_1	1	QSPI_IP_LUT_INSTR_STOP	QSPI_IP_LUT_PADS_1	0x0
2	FlsInstructionOperandPair_2	2	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_4	0x99

FlsLUT

Name: InitReset

FlsInstructionOperandPair*

Index	Name	Pair Index	Fls LUT Instruction	Fls LUT Pad	Fls LUT Operand
0	FlsInstructionOperandPair_0	0	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_1	0x66
1	FlsInstructionOperandPair_1	1	QSPI_IP_LUT_INSTR_STOP	QSPI_IP_LUT_PADS_1	0x0
2	FlsInstructionOperandPair_2	2	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_1	0x99

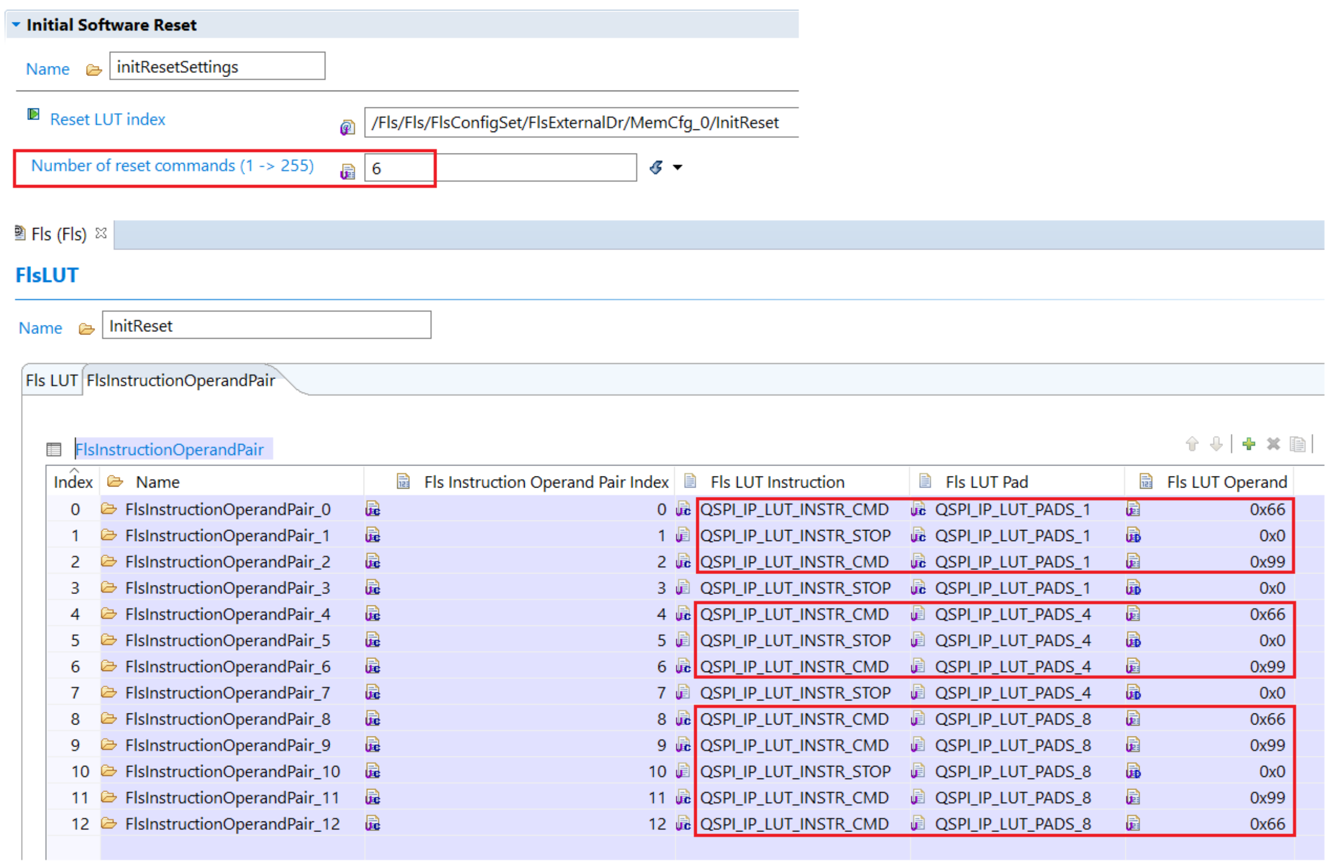
Note

- The number of reset commands is the number of sequences needed by the reset operation, separated by a stop phase
- A stop phase will be inserted automatically at the end of each command sequence

The **Initial Software Reset** procedure applies only at driver intialization. It might be different from the normal reset command, depending on the initial state of the flash. If not, set the same as reset command. In the above example, the memory device works in quad mode (4S-4S-4S), hence we need the reset commands in quad mode.

- The **Initial Software Reset** feature is useful in case we do not know the current state of the device memory (for example when bootrom leaves the memory in a certain state), and we need a reset sequence to bring it to the default state before performing initialization.

Here is an example of a combination of reset command sets (in three modes: SPI, QPI and OPI) to force memory device into its default state:



Initial Software Reset

Name: initResetSettings

Reset LUT index: /Fls/Fls/FlsConfigSet/FlsExternalDr/MemCfg_0/InitReset

Number of reset commands (1 -> 255): 6

Fls (Fls)

FlsLUT

Name: InitReset

Fls LUT | FlsInstructionOperandPair

FlsInstructionOperandPair

Index	Name	Fls Instruction Operand Pair Index	Fls LUT Instruction	Fls LUT Pad	Fls LUT Operand
0	FlsInstructionOperandPair_0	0	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_1	0x66
1	FlsInstructionOperandPair_1	1	QSPI_IP_LUT_INSTR_STOP	QSPI_IP_LUT_PADS_1	0x0
2	FlsInstructionOperandPair_2	2	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_1	0x99
3	FlsInstructionOperandPair_3	3	QSPI_IP_LUT_INSTR_STOP	QSPI_IP_LUT_PADS_1	0x0
4	FlsInstructionOperandPair_4	4	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_4	0x66
5	FlsInstructionOperandPair_5	5	QSPI_IP_LUT_INSTR_STOP	QSPI_IP_LUT_PADS_4	0x0
6	FlsInstructionOperandPair_6	6	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_4	0x99
7	FlsInstructionOperandPair_7	7	QSPI_IP_LUT_INSTR_STOP	QSPI_IP_LUT_PADS_4	0x0
8	FlsInstructionOperandPair_8	8	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_8	0x66
9	FlsInstructionOperandPair_9	9	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_8	0x99
10	FlsInstructionOperandPair_10	10	QSPI_IP_LUT_INSTR_STOP	QSPI_IP_LUT_PADS_8	0x0
11	FlsInstructionOperandPair_11	11	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_8	0x99
12	FlsInstructionOperandPair_12	12	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_8	0x66

Note

- Fls_Cancel() will use **Software Reset** to abort the on-going write/erase operation
- This action causes loss of synchronization between QuadSPI controller and memory device (for example when working in DOPI mode with external DQS)
- The next section will describe the solution to deal with this situation

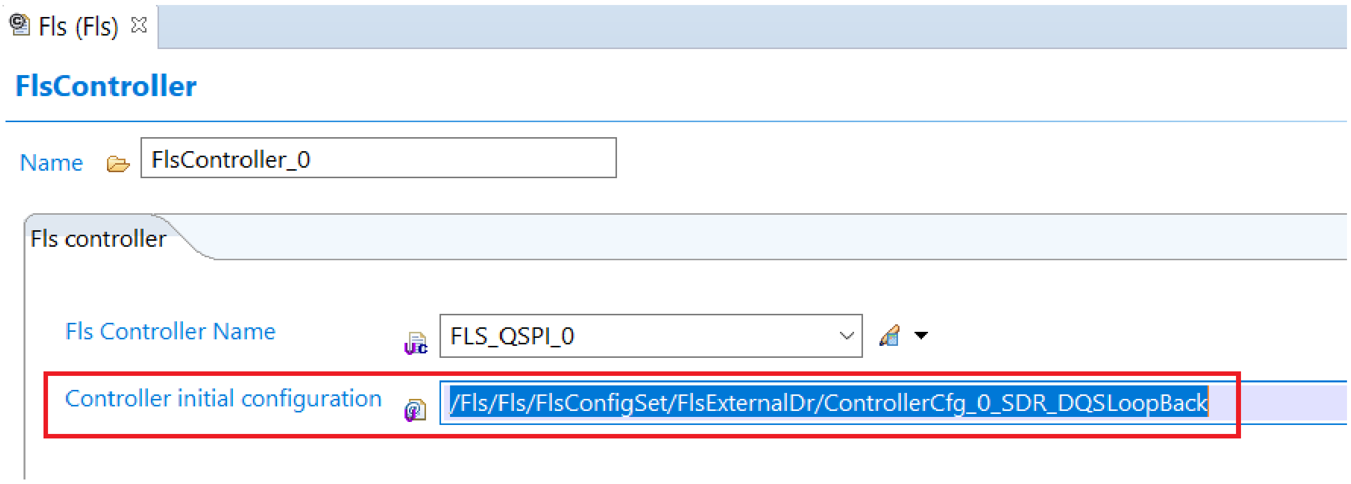
3.2.3.2.2 Controller configuration

There are several controller configuration points in the driver:

#	Controller configuration point	Location	Description
1	Controller initial configuration	FlsController	Configure QuadSPI controller
2	Controller configuration	MemCfg (operations list)	Re-configure QuadSPI controller during memory device initialization
			E.g: switch the controller to External DQS after activating DOPI mode
3	Configure controller on flash Init	MemCfg	Re-configure QuadSPI controller when resetting the memory device
			E.g: switch the controller to initial configuration before re-init the memory device

3.2.3.2.2.1 Controller initial configuration

This is the first initialization point which will be done once by Fls_Init():


Fls (Fls)

FlsController

Name: FlsController_0

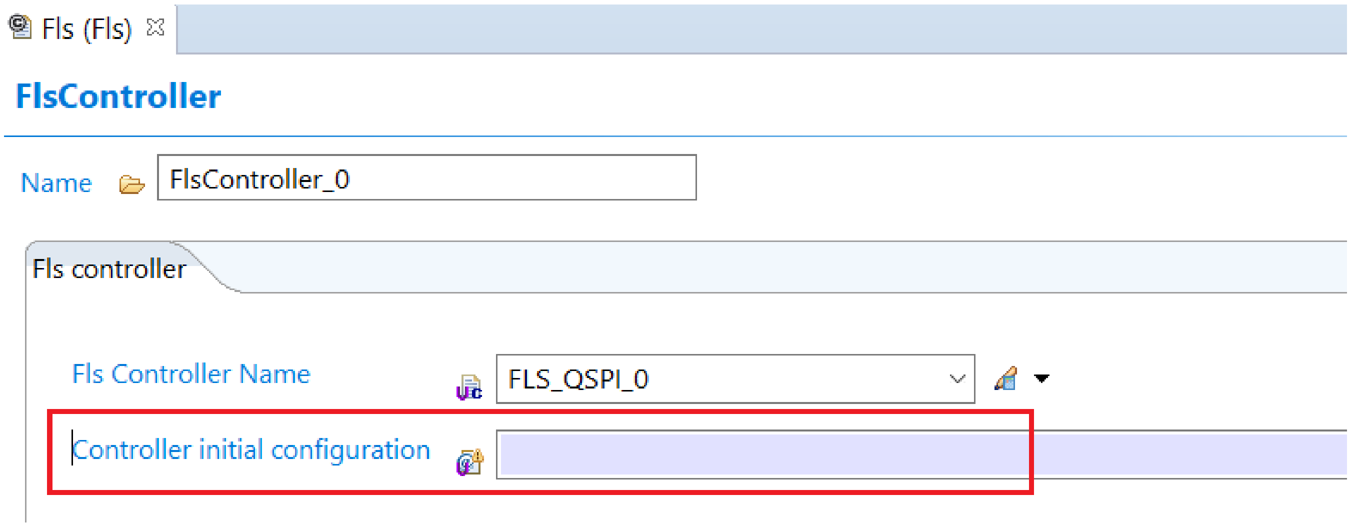
Fls controller

Fls Controller Name: FLS_QSPI_0

Controller initial configuration: /Fls/Fls/FlsConfigSet/FlsExternalDr/ControllerCfg_0_SDR_DQSLoopBack

This step can be skipped by leaving the reference node blank, the purpose is:

- Reuse the existing QuadSPI settings by the boot code
- Or in multicore context where one core performs the initialization for the others


Fls (Fls)

FlsController

Name: FlsController_0

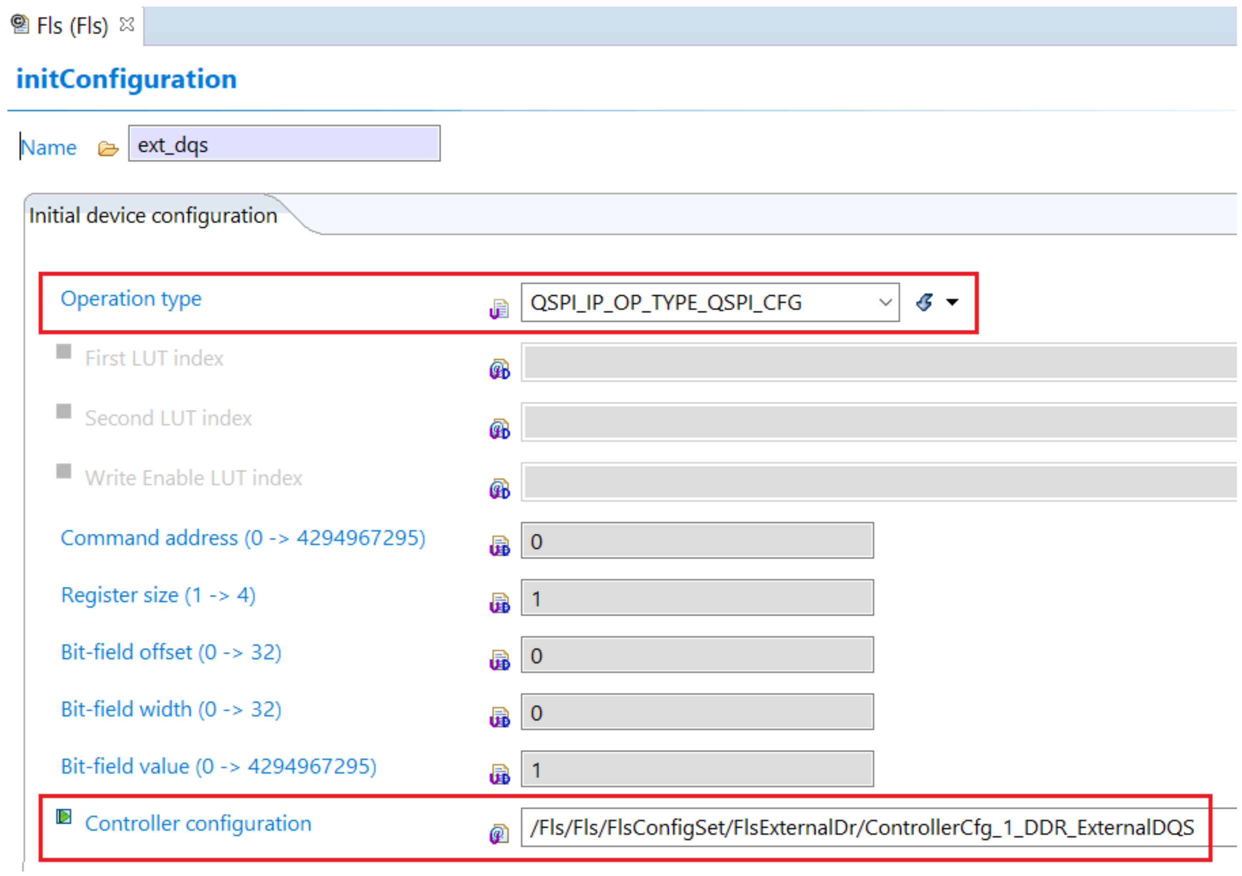
Fls controller

Fls Controller Name: FLS_QSPI_0

Controller initial configuration:

3.2.3.2.2.2 Controller configuration

The second initialization point is in the list of operations of **MemCfg**. This step is needed after we configure the device memory to another mode that is no longer compatible with the controller configuration point #1 (E.g: after activating DOPI mode)



Fls (Fls)

initConfiguration

Name: ext_dqs

Initial device configuration

Operation type: QSPI_IP_OP_TYPE_QSPI_CFG

First LUT index

Second LUT index

Write Enable LUT index

Command address (0 -> 4294967295): 0

Register size (1 -> 4): 1

Bit-field offset (0 -> 32): 0

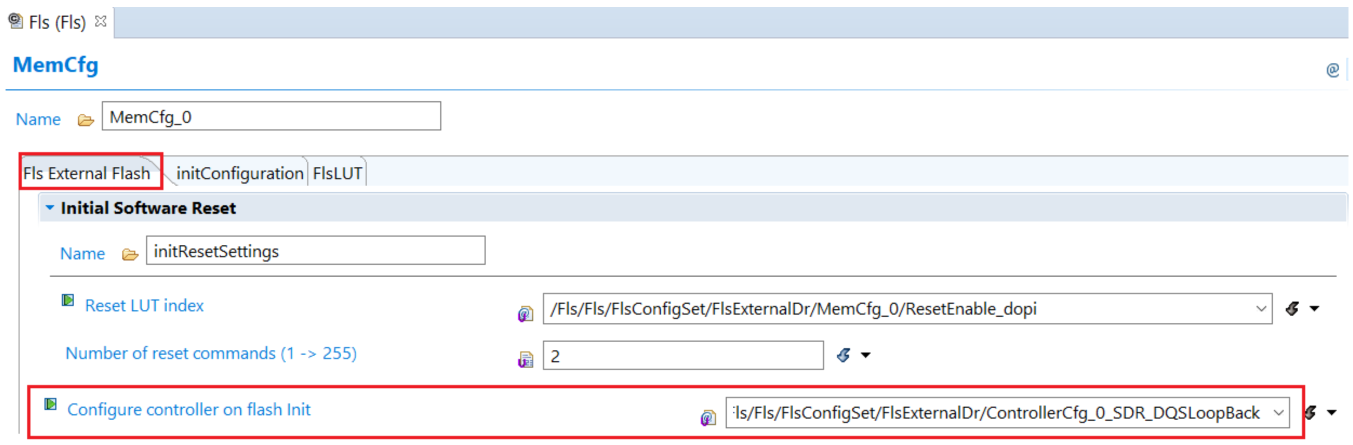
Bit-field width (0 -> 32): 0

Bit-field value (0 -> 4294967295): 1

Controller configuration: /Fls/Fls/FlsConfigSet/FlsExternalDr/ControllerCfg_1_DDR_ExternalDQS

3.2.3.2.2.3 Configure controller on flash Init

The third configuration point is at the end of **MemCfg**. This step is needed when resetting the memory device, then we have to re-configure the controller to a mode that is compatible with the new state of memory device after reset. (E.g: when executing the reset sequence in DOPI mode)



Fls (Fls)

MemCfg

Name: MemCfg_0

Fls External Flash | initConfiguration | FlsLUT

Initial Software Reset

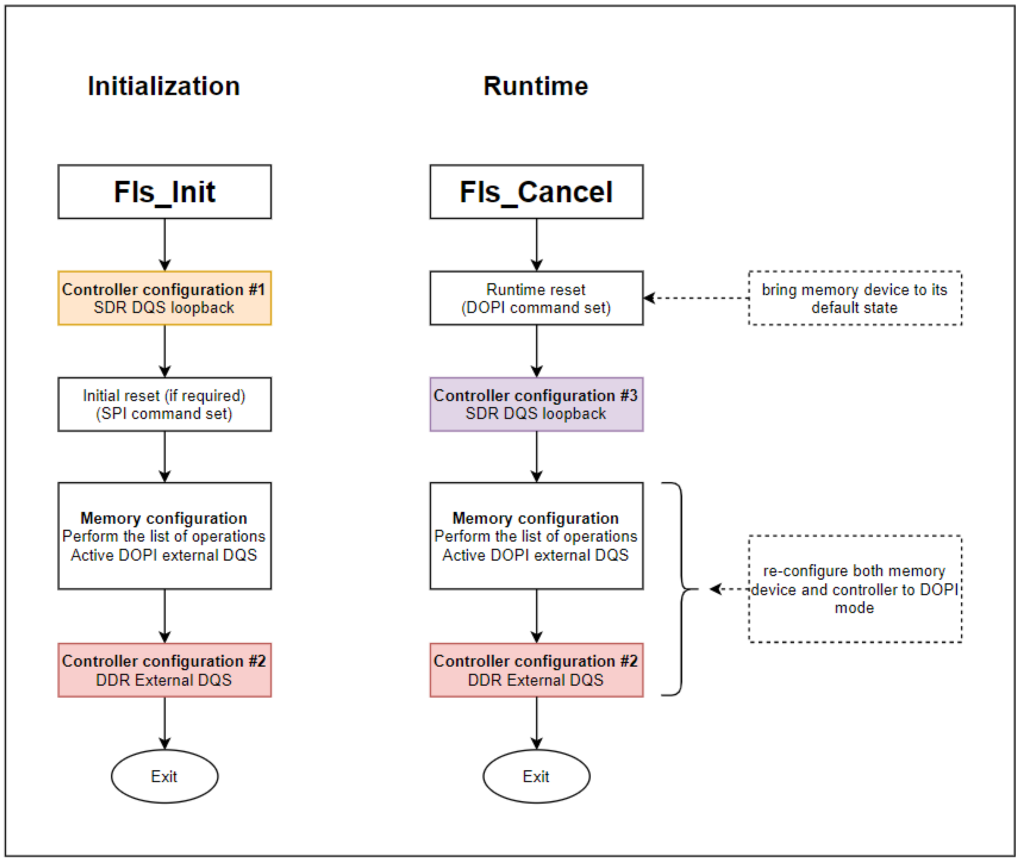
Name: initResetSettings

Reset LUT index: /Fls/Fls/FlsConfigSet/FlsExternalDr/MemCfg_0/ResetEnable_dopi

Number of reset commands (1 -> 255): 2

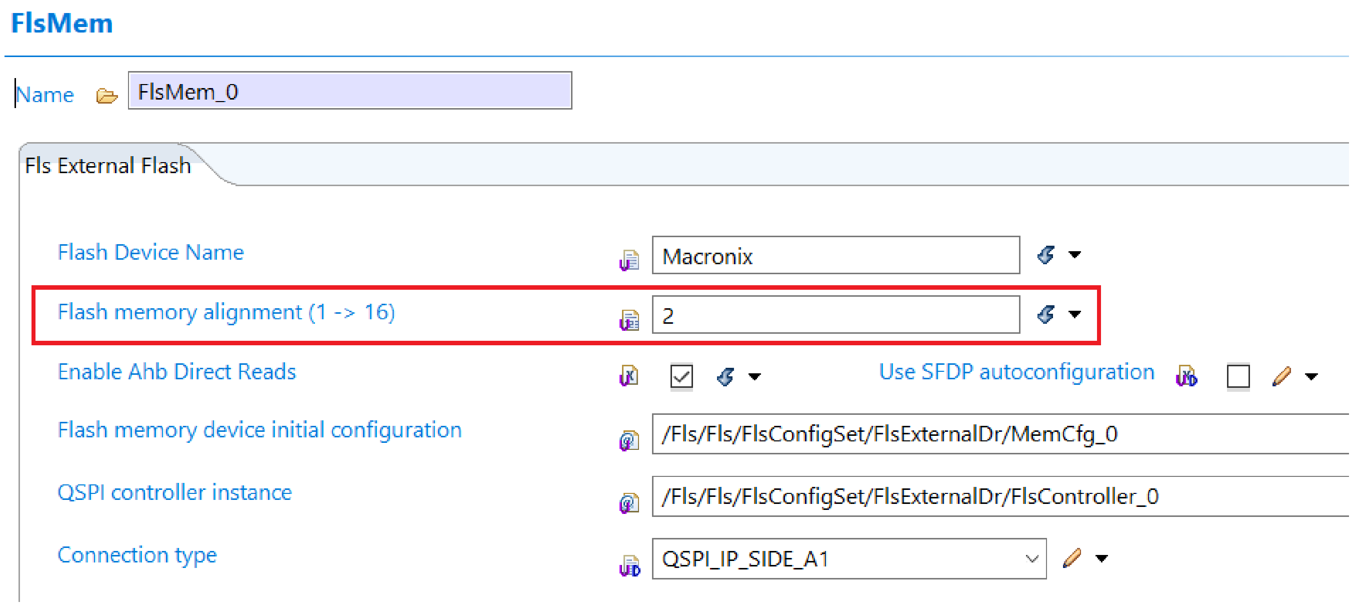
Configure controller on flash Init: ls/Fls/FlsConfigSet/FlsExternalDr/ControllerCfg_0_SDR_DQSLoopBack

Below is the complete code flow (initialization time and runtime) for both QuadSPI controller and memory device to work in Double data rate - Octal I/O mode (DOPI).


Initialization
Runtime
Fls_Init
Controller configuration #1
SDR DQS loopback
Initial reset (if required)
(SPI command set)
Memory configuration
Perform the list of operations
Active DOPI external DQS
Controller configuration #2
DDR External DQS
Exit
Fls_Cancel
Runtime reset
(DOPI command set)
bring memory device to its default state
Controller configuration #3
SDR DQS loopback
Memory configuration
Perform the list of operations
Active DOPI external DQS
re-configure both memory device and controller to DOPI mode
Controller configuration #2
DDR External DQS
Exit

3.2.3.2.3 Read/Write from unaligned addresses

Due to the nature of DDR protocol, both the starting address must be even address and data byte number must be even. Fls driver supports a feature to allow users to read/write with odd addresses and odd data length in DOPI mode, simply by setting the memory alignment value to 2 in the **FlsMem** configuration:



FlsMem

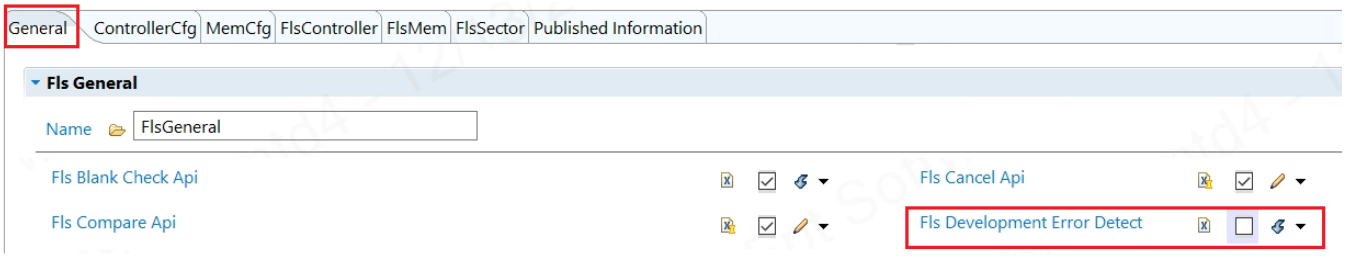
Name: FlsMem_0

Fls External Flash

Flash Device Name	Macronix
Flash memory alignment (1 -> 16)	2
Enable Ahb Direct Reads	✓
Use SFDP autoconfiguration	☐
Flash memory device initial configuration	/Fls/Fls/FlsConfigSet/FlsExternalDr/MemCfg_0
QSPI controller instance	/Fls/Fls/FlsConfigSet/FlsExternalDr/FlsController_0
Connection type	QSPI_IP_SIDE_A1

Note

- For write operation, driver will send extra data with FFh to overwrite the overlapping memory area
- Users need to disable the development error detection feature in order to bypass the flash page boundary alignment checks:



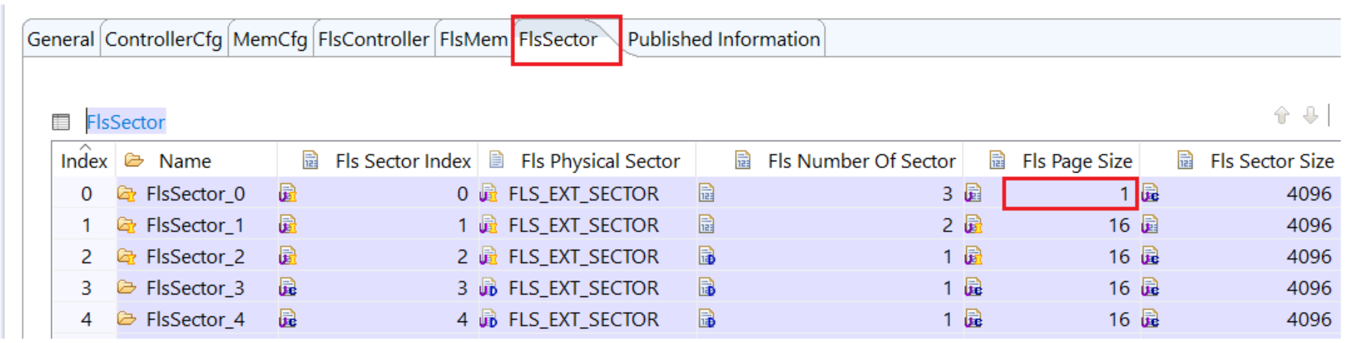
General | ControllerCfg | MemCfg | FlsController | FlsMem | FlsSector | Published Information

Fls General

Name: FlsGeneral

Fls Blank Check Api	✓	Fls Cancel Api	✓
Fls Compare Api	✓	Fls Development Error Detect	☐

- Or configure the size of flash page boundary to **1** to meet that requirement



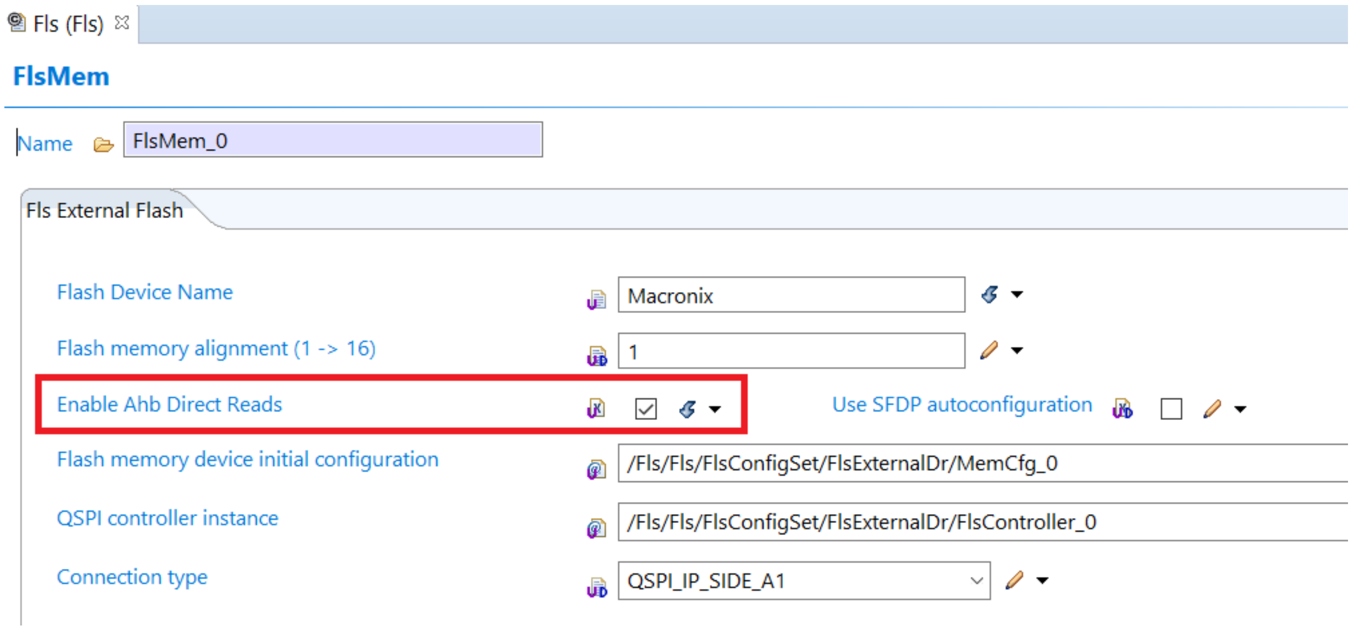
General | ControllerCfg | MemCfg | FlsController | FlsMem | FlsSector | Published Information

FlsSector

Index	Name	Fls Sector Index	Fls Physical Sector	Fls Number Of Sector	Fls Page Size	Fls Sector Size
0	FlsSector_0	0	FLS_EXT_SECTOR	3	1	4096
1	FlsSector_1	1	FLS_EXT_SECTOR	2	16	4096
2	FlsSector_2	2	FLS_EXT_SECTOR	1	16	4096
3	FlsSector_3	3	FLS_EXT_SECTOR	1	16	4096
4	FlsSector_4	4	FLS_EXT_SECTOR	1	16	4096

3.2.3.2.4 AHB read

Users can enable the option **AHBReadEnable** in **FlsMem** to use the AHB read feature, this allows application can read directly through Flash memory devices address mapping (QuadSPI's AHB region):



Fls (Fls)

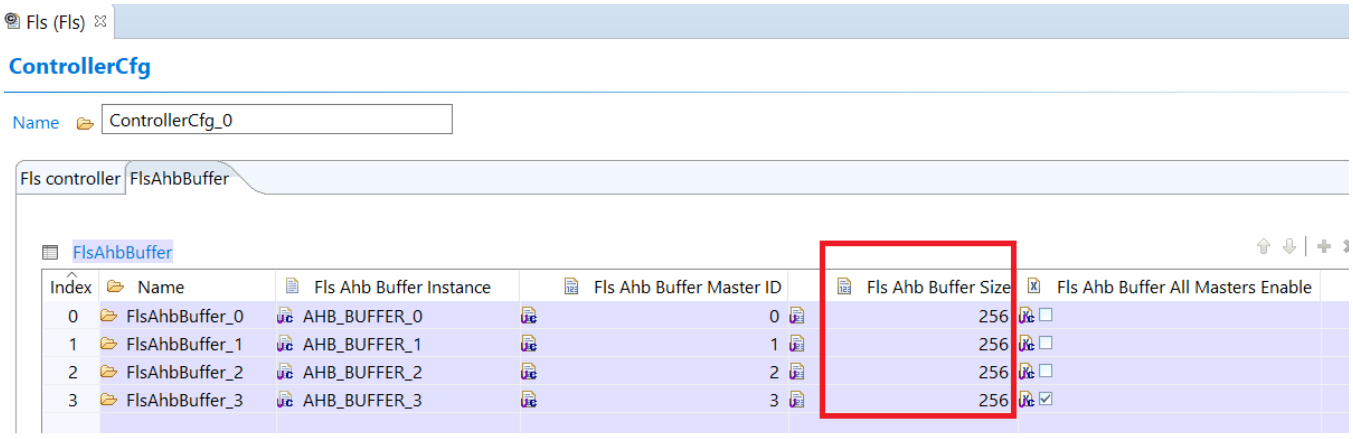
FlsMem

Name: FlsMem_0

Fls External Flash

Flash Device Name	Macronix
Flash memory alignment (1 -> 16)	1
Enable Ahb Direct Reads	✓
Use SFDP autoconfiguration	☐
Flash memory device initial configuration	/Fls/Fls/FlsConfigSet/FlsExternalDr/MemCfg_0
QSPI controller instance	/Fls/Fls/FlsConfigSet/FlsExternalDr/FlsController_0
Connection type	QSPI_IP_SIDE_A1

Besides, users need to configure the AHB buffers (master IDs and sizes):


Fls (Fls)

ControllerCfg

Name: ControllerCfg_0

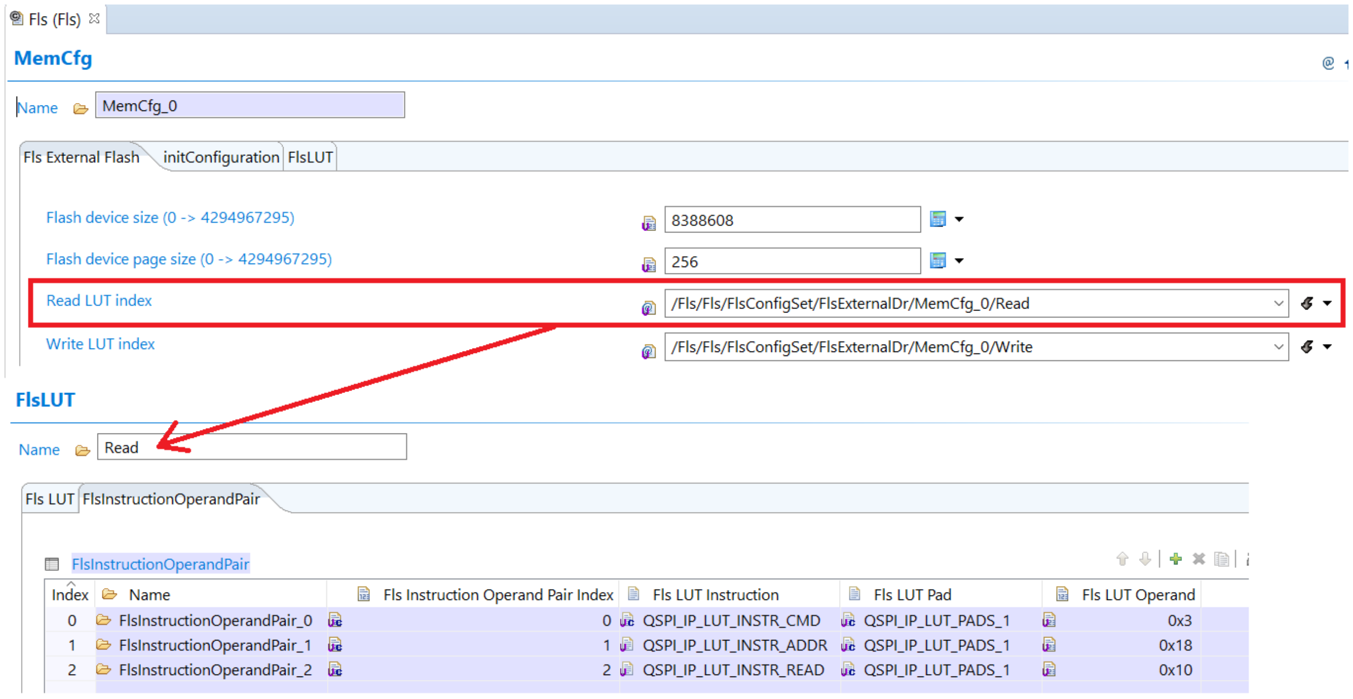
FlsAhbBuffer

Index	Name	Fls Ahb Buffer Instance	Fls Ahb Buffer Master ID	Fls Ahb Buffer Size	Fls Ahb Buffer All Masters Enable
0	FlsAhbBuffer_0	AHB_BUFFER_0	0	256	☐
1	FlsAhbBuffer_1	AHB_BUFFER_1	1	256	☐
2	FlsAhbBuffer_2	AHB_BUFFER_2	2	256	☐
3	FlsAhbBuffer_3	AHB_BUFFER_3	3	256	☑

Note

- The driver will configure AHB transfer sizes to match the buffer sizes
- **Qspi_Ip_ControllerGetStatus** can be used to wait for AHB commands complete to avoid conflict with subsequent IP commands

The LUT sequence of IP command read will be used for AHB command read:


Fls (Fls)

MemCfg

Name: MemCfg_0

Flash device size (0 -> 4294967295): 8388608

Flash device page size (0 -> 4294967295): 256

Read LUT index: /Fls/Fls/FlsConfigSet/FlsExternalDr/MemCfg_0/Read

Write LUT index: /Fls/Fls/FlsConfigSet/FlsExternalDr/MemCfg_0/Write

FlsLUT

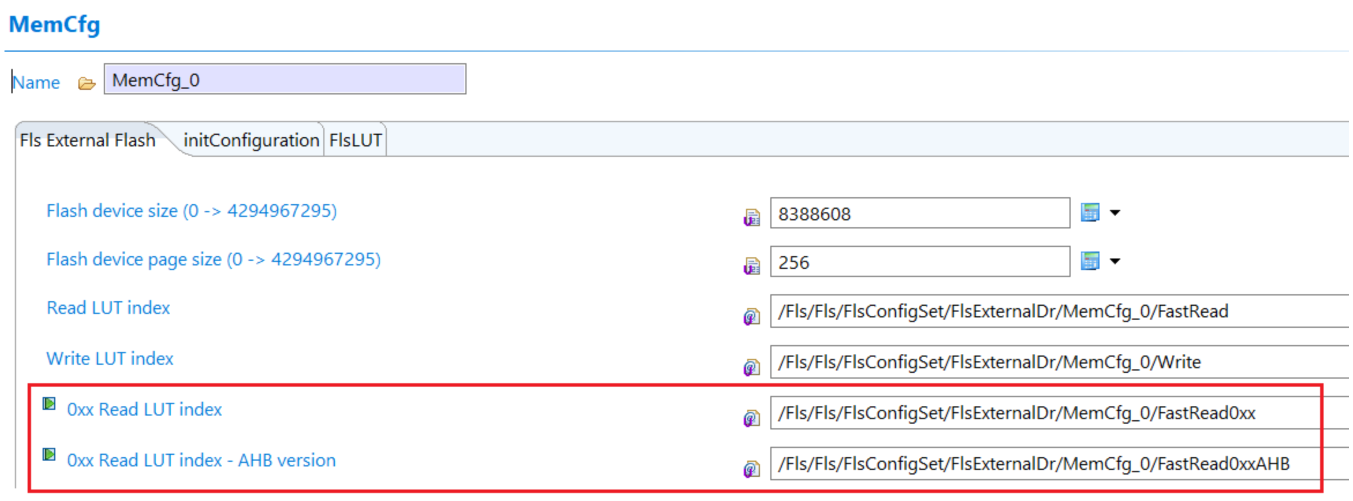
Name: Read

FlsInstructionOperandPair

Index	Name	Fls Instruction Operand Pair Index	Fls LUT Instruction	Fls LUT Pad	Fls LUT Operand
0	FlsInstructionOperandPair_0	0	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_1	0x3
1	FlsInstructionOperandPair_1	1	QSPI_IP_LUT_INSTR_ADDR	QSPI_IP_LUT_PADS_1	0x18
2	FlsInstructionOperandPair_2	2	QSPI_IP_LUT_INSTR_READ	QSPI_IP_LUT_PADS_1	0x10

3.2.3.2.5 Performance enhanced mode

The QuadSPI driver supports Continuous Read mode (0-X-X mode - no command for read instructions) which is implemented in some serial Flash memories:



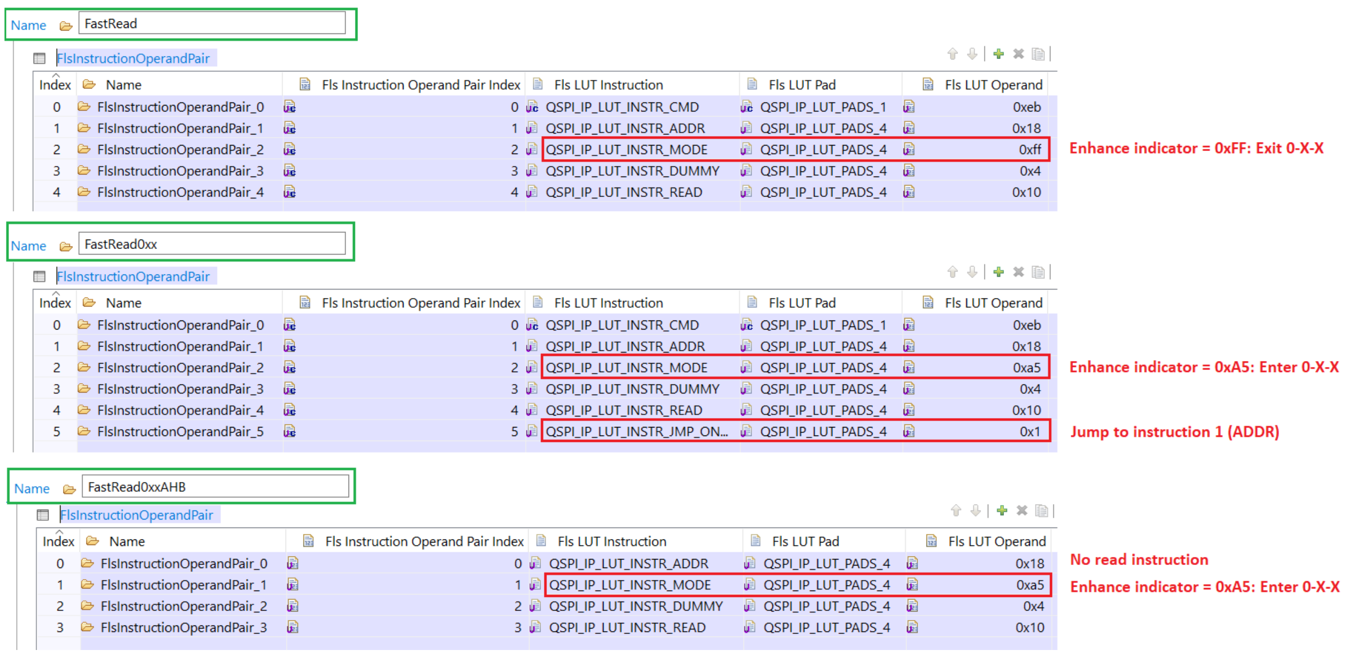
MemCfg

Name: MemCfg_0

Fls External Flash | initConfiguration | FlsLUT

Flash device size (0 -> 4294967295)	8388608
Flash device page size (0 -> 4294967295)	256
Read LUT index	/Fls/Fls/FlsConfigSet/FlsExternalDr/MemCfg_0/FastRead
Write LUT index	/Fls/Fls/FlsConfigSet/FlsExternalDr/MemCfg_0/Write
0xx Read LUT index	/Fls/Fls/FlsConfigSet/FlsExternalDr/MemCfg_0/FastRead0xx
0xx Read LUT index - AHB version	/Fls/Fls/FlsConfigSet/FlsExternalDr/MemCfg_0/FastRead0xxAHB

There are two types of command, one for IP and one for AHB operations. Below is the example:



Name: FastRead

FlsInstructionOperandPair

Index	Name	Fls Instruction Operand Pair Index	Fls LUT Instruction	Fls LUT Pad	Fls LUT Operand
0	FlsInstructionOperandPair_0	0	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_1	0xeb
1	FlsInstructionOperandPair_1	1	QSPI_IP_LUT_INSTR_ADDR	QSPI_IP_LUT_PADS_4	0x18
2	FlsInstructionOperandPair_2	2	QSPI_IP_LUT_INSTR_MODE	QSPI_IP_LUT_PADS_4	0xff
3	FlsInstructionOperandPair_3	3	QSPI_IP_LUT_INSTR_DUMMY	QSPI_IP_LUT_PADS_4	0x4
4	FlsInstructionOperandPair_4	4	QSPI_IP_LUT_INSTR_READ	QSPI_IP_LUT_PADS_4	0x10

Enhance indicator = 0xFF: Exit 0-X-X

Name: FastRead0xx

FlsInstructionOperandPair

Index	Name	Fls Instruction Operand Pair Index	Fls LUT Instruction	Fls LUT Pad	Fls LUT Operand
0	FlsInstructionOperandPair_0	0	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_1	0xeb
1	FlsInstructionOperandPair_1	1	QSPI_IP_LUT_INSTR_ADDR	QSPI_IP_LUT_PADS_4	0x18
2	FlsInstructionOperandPair_2	2	QSPI_IP_LUT_INSTR_MODE	QSPI_IP_LUT_PADS_4	0xa5
3	FlsInstructionOperandPair_3	3	QSPI_IP_LUT_INSTR_DUMMY	QSPI_IP_LUT_PADS_4	0x4
4	FlsInstructionOperandPair_4	4	QSPI_IP_LUT_INSTR_READ	QSPI_IP_LUT_PADS_4	0x10
5	FlsInstructionOperandPair_5	5	QSPI_IP_LUT_INSTR_JMP_ON...	QSPI_IP_LUT_PADS_4	0x1

Enhance indicator = 0xA5: Enter 0-X-X

Jump to instruction 1 (ADDR)

Name: FastRead0xxAHB

FlsInstructionOperandPair

Index	Name	Fls Instruction Operand Pair Index	Fls LUT Instruction	Fls LUT Pad	Fls LUT Operand
0	FlsInstructionOperandPair_0	0	QSPI_IP_LUT_INSTR_ADDR	QSPI_IP_LUT_PADS_4	0x18
1	FlsInstructionOperandPair_1	1	QSPI_IP_LUT_INSTR_MODE	QSPI_IP_LUT_PADS_4	0xa5
2	FlsInstructionOperandPair_2	2	QSPI_IP_LUT_INSTR_DUMMY	QSPI_IP_LUT_PADS_4	0x4
3	FlsInstructionOperandPair_3	3	QSPI_IP_LUT_INSTR_READ	QSPI_IP_LUT_PADS_4	0x10

No read instruction

Enhance indicator = 0xA5: Enter 0-X-X

How they work:

1. (Optional) Call the **Qspi_Ip_AhbReadEnable** to enable AHB operation
2. Call the **Qspi_Ip_Enter0XX** to switch to 0-X-X read command sets, driver will perform a dummy read to activate 0-X-X mode

3. Call the **Qspi_Ip_Read** to read data from flash memory without the send of the instruction code

4. (Optional) Access the QuadSPI's AHB region to read data directly, **Qspi_Ip_ControllerGetStatus** can be used to wait for AHB commands complete to avoid conflict with subsequent IP commands

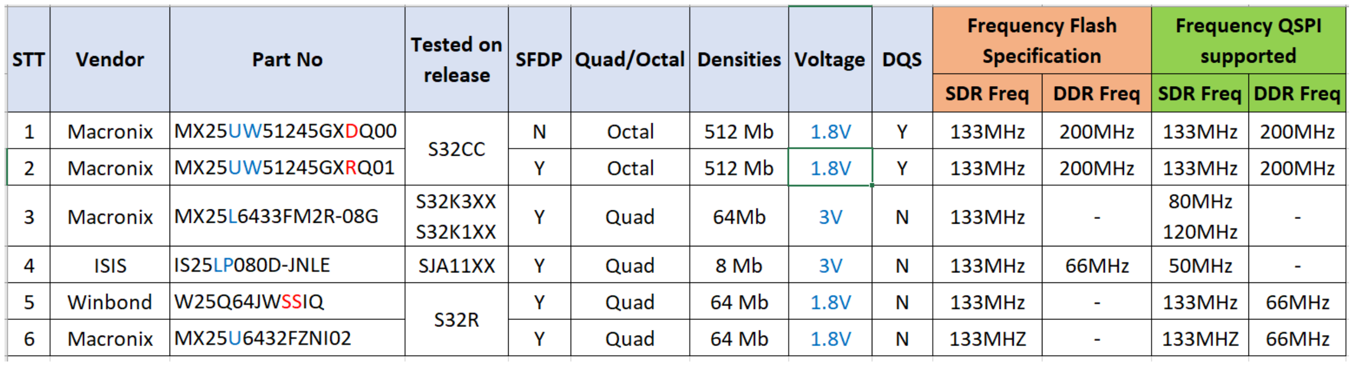
5. Call the **Qspi_Ip_Exit0XX** to disable 0-X-X mode and switch back to normal read command sets

Note

Qspi_Ip_ClearIpSeqPointer() and **Qspi_Ip_ClearAHBSeqPointer()** can be useful for devices which support burst modes for enhancing performance

3.2.3.3 Supported memories

The following external device memories were tested by the Fls driver:



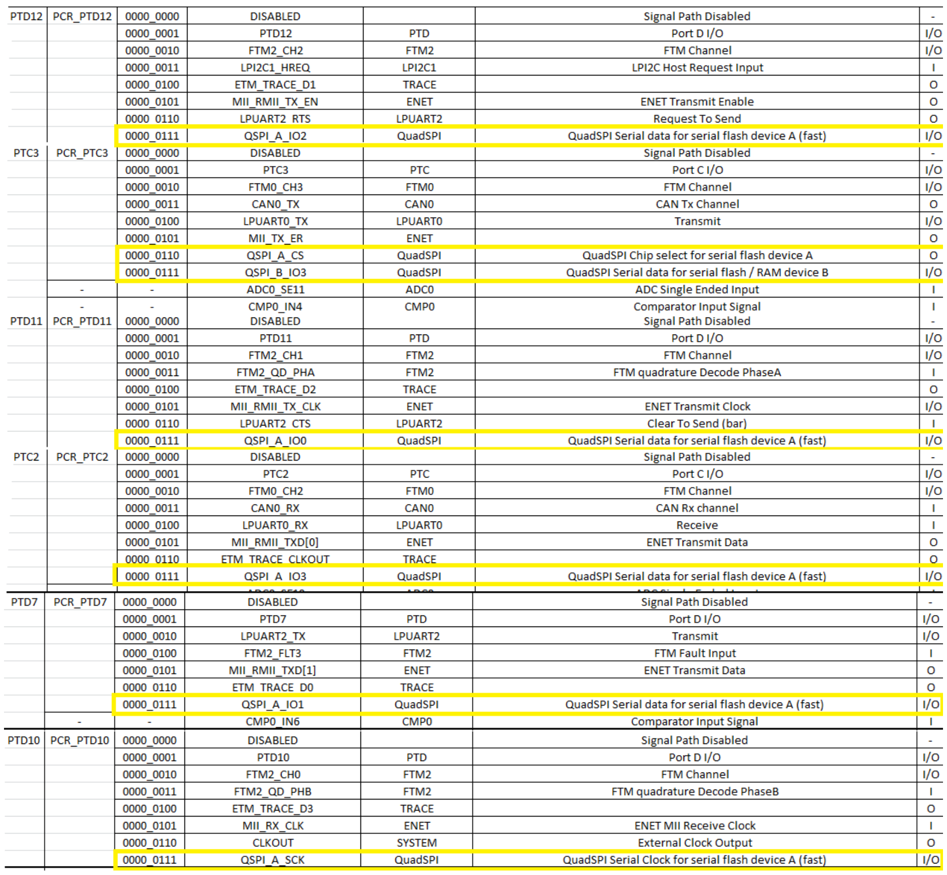
STT	Vendor	Part No	Tested on release	SFDP	Quad/Octal	Densities	Voltage	DQS	Frequency Flash Specification		Frequency QSPI supported	
									SDR Freq	DDR Freq	SDR Freq	DDR Freq
1	Macronix	MX25UW51245GXDQ00	S32CC	N	Octal	512 Mb	1.8V	Y	133MHz	200MHz	133MHz	200MHz
2	Macronix	MX25UW51245GXRQ01		Y	Octal	512 Mb	1.8V	Y	133MHz	200MHz	133MHz	200MHz
3	Macronix	MX25L6433FM2R-08G	S32K3XX S32K1XX	Y	Quad	64Mb	3V	N	133MHz	-	80MHz 120MHz	-
4	ISIS	IS25LP080D-JNLE	SJA11XX	Y	Quad	8 Mb	3V	N	133MHz	66MHz	50MHz	-
5	Winbond	W25Q64JWSSIQ	S32R	Y	Quad	64 Mb	1.8V	N	133MHz	-	133MHz	66MHz
6	Macronix	MX25U6432FZNI02		Y	Quad	64 Mb	1.8V	N	133MHZ	-	133MHZ	66MHz

3.3 Hardware Resources

3.3.1 External flash

The Quad Serial Peripheral Interface (QuadSPI) block acts as an interface to external serial flash device. It supports SDR and HyperRAM modes upto 4 and 8 bidirectional data lines respectively.

For external flash: Configure pins for QSPI pin mux, driver strength enable, pull up enabled.



PTD12	PCR_PTD12	0000_0000	DISABLED		Signal Path Disabled	-
		0000_0001	PTD12	PTD	Port D I/O	I/O
		0000_0010	FTM2_CH2	FTM2	FTM Channel	I/O
		0000_0011	LPI2C1_HREQ	LPI2C1	LPI2C Host Request Input	I
		0000_0100	ETM_TRACE_D1	TRACE		O
		0000_0101	MII_RMII_TX_EN	ENET	ENET Transmit Enable	O
		0000_0110	LPUART2_RTS	LPUART2	Request To Send	O
		0000_0111	QSPI_A_IO2	QuadSPI	QuadSPI Serial data for serial flash device A (fast)	I/O
PTC3	PCR_PTC3	0000_0000	DISABLED		Signal Path Disabled	-
		0000_0001	PTC3	PTC	Port C I/O	I/O
		0000_0010	FTM0_CH3	FTM0	FTM Channel	I/O
		0000_0011	CAN0_TX	CAN0	CAN Tx Channel	O
		0000_0100	LPUART0_TX	LPUART0	Transmit	I/O
		0000_0101	MII_TX_ER	ENET		O
		0000_0110	QSPI_A_CS	QuadSPI	QuadSPI Chip select for serial flash device A	O
		0000_0111	QSPI_B_IO3	QuadSPI	QuadSPI Serial data for serial flash / RAM device B	I/O
	-	-	ADC0_SE11	ADC0	ADC Single Ended Input	I
	-	-	CMP0_IN4	CMP0	Comparator Input Signal	I
PTD11	PCR_PTD11	0000_0000	DISABLED		Signal Path Disabled	-
		0000_0001	PTD11	PTD	Port D I/O	I/O
		0000_0010	FTM2_CH1	FTM2	FTM Channel	I/O
		0000_0011	FTM2_QD_PHA	FTM2	FTM quadrature Decode PhaseA	I
		0000_0100	ETM_TRACE_D2	TRACE		O
		0000_0101	MII_RMII_TX_CLK	ENET	ENET Transmit Clock	I/O
		0000_0110	LPUART2_CTS	LPUART2	Clear To Send (bar)	I
		0000_0111	QSPI_A_IO0	QuadSPI	QuadSPI Serial data for serial flash device A (fast)	I/O
PTC2	PCR_PTC2	0000_0000	DISABLED		Signal Path Disabled	-
		0000_0001	PTC2	PTC	Port C I/O	I/O
		0000_0010	FTM0_CH2	FTM0	FTM Channel	I/O
		0000_0011	CAN0_RX	CAN0	CAN Rx channel	I
		0000_0100	LPUART0_RX	LPUART0	Receive	I
		0000_0101	MII_RMII_TXD[0]	ENET	ENET Transmit Data	O
		0000_0110	ETM_TRACE_CLKOUT	TRACE		O
		0000_0111	QSPI_A_IO3	QuadSPI	QuadSPI Serial data for serial flash device A (fast)	I/O
PTD7	PCR_PTD7	0000_0000	DISABLED		Signal Path Disabled	-
		0000_0001	PTD7	PTD	Port D I/O	I/O
		0000_0010	LPUART2_TX	LPUART2	Transmit	I/O
		0000_0100	FTM2_FLT3	FTM2	FTM Fault Input	I
		0000_0101	MII_RMII_TXD[1]	ENET	ENET Transmit Data	O
		0000_0110	ETM_TRACE_D0	TRACE		O
		0000_0111	QSPI_A_IO1	QuadSPI	QuadSPI Serial data for serial flash device A (fast)	I/O
	-	-	CMP0_IN6	CMP0	Comparator Input Signal	I
PTD10	PCR_PTD10	0000_0000	DISABLED		Signal Path Disabled	-
		0000_0001	PTD10	PTD	Port D I/O	I/O
		0000_0010	FTM2_CH0	FTM2	FTM Channel	I/O
		0000_0011	FTM2_QD_PHB	FTM2	FTM quadrature Decode PhaseB	I
		0000_0100	ETM_TRACE_D3	TRACE		O
		0000_0101	MII_RX_CLK	ENET	ENET MII Receive Clock	I
		0000_0110	CLKOUT	SYSTEM	External Clock Output	O
		0000_0111	QSPI_A_SCK	QuadSPI	QuadSPI Serial Clock for serial flash device A (fast)	I/O

Note:

- S32K148 (Except S32K148_lqfp100 has none) has one instance of QuadSPI. Other products in the S32K1xx series do not have QuadSPI.
- The following are not supported:
  - AHB Write
  - Data learning feature
  - Breakpoint and Watchpoint memory regions
  - QuadSPI in 100-pin LQFP

3.3.2 S32K1 Flash Banks/Arrays, Sectors details

- The sizes of flash memory types on the chip are:

Chips	Program Flash	Data Flash (FlexNVM)
S32K116	128 KB (sector size 2k)	32 KB (sector size 2k)
S32K118	256 KB (sector size 2k)	32 KB (sector size 2k)
S32K142	256 KB (sector size 2k)	64 KB (sector size 2k)
S32K142W	256 KB (sector size 4k)	64 KB (sector size 2k)

Chips	Program Flash	Data Flash (FlexNVM)
S32K144	512 KB (sector size 4k)	64 KB (sector size 2k)
S32K144W	512 KB (sector size 4k)	64 KB (sector size 2k)
S32K146	1 MB (sector size 4k)	64 KB (sector size 2k)
S32K148	1.5 MB (sector size 4k)	512 KB (sector size 4k)

- For S32K116: has 128 KBytes of code flash (program flash) and 32 KBytes of data flash (FlexNVM)
  - 128K P Flash (each sector is 2K so 128K/2K = 64 sectors)
  - 32K D Flash (each sector is 2K so 32K/2K = 16 sectors)
  - There are 2 blocks (read partitions):
    - P Flash: Block 0 (128K)
    - D Flash: Block 1 (32K)

Sector name	Sector Size (KB)
FLS_DATA_ARRAY_0_BLOCK_1_S000	2
...	...
FLS_DATA_ARRAY_0_BLOCK_1_S015	2
FLS_CODE_ARRAY_0_BLOCK_0_S000	2
...	...
FLS_CODE_ARRAY_0_BLOCK_0_S063	2

- For S32K118: has 256 KBytes of code flash (program flash) and 32 KBytes of data flash (FlexNVM)
  - 256K P Flash (each sector is 2K so 256K/2K = 128 sectors)
  - 32K D Flash (each sector is 2K so 32K/2K = 16 sectors)
  - There are 2 blocks (read partitions):
    - P Flash: Block 0 (256K)
    - D Flash: Block 1 (32K)

Sector name	Sector Size (KB)
FLS_DATA_ARRAY_0_BLOCK_1_S000	2
...	...
FLS_DATA_ARRAY_0_BLOCK_1_S015	2
FLS_CODE_ARRAY_0_BLOCK_0_S000	2
...	...
FLS_CODE_ARRAY_0_BLOCK_0_S127	2

- For S32K142: has 256 KBytes of code flash (program flash) and 64 KBytes of data flash (FlexNVM)
  - 256K P Flash (each sector is 2K so 256K/2K = 128 sectors)
  - 64K D Flash (each sector is 2K so 64K/2K = 32 sectors)
  - There are 2 blocks (read partitions):

        * P Flash: Block 0 (256K)
        * D Flash: Block 1 (64K)

Sector name	Sector Size (KB)
FLS_DATA_ARRAY_0_BLOCK_1_S000	2
...	...
FLS_DATA_ARRAY_0_BLOCK_1_S031	2
FLS_CODE_ARRAY_0_BLOCK_0_S000	2
...	...
FLS_CODE_ARRAY_0_BLOCK_0_S127	2

- For S32K142W: has 256 KBytes of code flash (program flash) and 64 KBytes of data flash (FlexNVM)
    - 256K P Flash (each sector is 4K so 256K/4K = 64 sectors)
    - 64K D Flash (each sector is 2K so 64K/2K = 32 sectors)
    - There are 2 blocks (read partitions):
        * P Flash: Block 0 (256K)
        * D Flash: Block 1 (64K)

Sector name	Sector Size (KB)
FLS_DATA_ARRAY_0_BLOCK_1_S000	2
...	...
FLS_DATA_ARRAY_0_BLOCK_1_S031	2
FLS_CODE_ARRAY_0_BLOCK_0_S000	4
...	...
FLS_CODE_ARRAY_0_BLOCK_0_S063	4

- For S32K144: has 512 KBytes of code flash (program flash) and 64 KBytes of data flash (FlexNVM)
    - 512K P Flash (each sector is 4K so 512/4K = 128 sectors)
    - 64K D Flash (each sector is 2K so 64K/2K = 32 sectors)
    - There are 2 blocks (read partitions):
        * P Flash: Block 0 (512K)
        * D Flash: Block 1 (64K)

Sector name	Sector Size (KB)
FLS_DATA_ARRAY_0_BLOCK_1_S000	2
...	...
FLS_DATA_ARRAY_0_BLOCK_1_S031	2
FLS_CODE_ARRAY_0_BLOCK_0_S000	4
...	...
FLS_CODE_ARRAY_0_BLOCK_0_S127	4

- For S32K144W: has 512 KBytes of code flash (program flash) and 64 KBytes of data flash (FlexNVM)

  - 512K P Flash (each sector is 4K so 512/4K = 128 sectors)
  - 64K D Flash (each sector is 2K so 64K/2K = 32 sectors)
  - There are 2 blocks (read partitions):
    * P Flash: Block 0 (512K)
    * D Flash: Block 1 (64K)

Sector name	Sector Size (KB)
FLS_DATA_ARRAY_0_BLOCK_1_S000	2
...	...
FLS_DATA_ARRAY_0_BLOCK_1_S031	2
FLS_CODE_ARRAY_0_BLOCK_0_S000	4
...	...
FLS_CODE_ARRAY_0_BLOCK_0_S127	4

- For S32K146: has 1 MBytes of code flash (program flash) and 64 KBytes of data flash (FlexNVM)
  - 1M P Flash (each sector is 4K so 1M/4K = 256 sectors)
  - 64K D Flash (each sector is 2K so 64K/2K = 128 sectors)
  - There are 3 blocks (read partitions):
    * P Flash: Block 0 (512K), Block 1 (512K)
    * D Flash: Block 2 (64K)

Sector name	Sector Size (KB)
FLS_DATA_ARRAY_0_BLOCK_2_S000	2
...	...
FLS_DATA_ARRAY_0_BLOCK_2_S031	2
FLS_CODE_ARRAY_0_BLOCK_0_S000	4
...	...
FLS_CODE_ARRAY_0_BLOCK_1_S255	4

- For S32K148: has 1.5 MBytes of code flash (program flash) and 512 KBytes of data flash (FlexNVM)
  - 1.5M P Flash (each sector is 4K so 1.5M/4K = 384 sectors)
  - 512K D Flash (each sector is 4K so 512K/4K = 128 sectors)
  - There are 4 blocks (read partitions):
    * P Flash: Block 0 (512K), Block 1 (512K), Block 2 (512K)
    * D Flash: Block 3 (512K)

Sector name	Sector Size (KB)
FLS_DATA_ARRAY_0_BLOCK_3_S000	4
...	...
FLS_DATA_ARRAY_0_BLOCK_3_S127	4
FLS_CODE_ARRAY_0_BLOCK_0_S000	4
...	...
FLS_CODE_ARRAY_0_BLOCK_2_S383	4

3.4 Deviations from Requirements

The driver deviates from the AUTOSAR FLS Driver software specification in some places. There are also some additional requirements (on top of requirements detailed in AUTOSAR FLS Driver software specification) which need to be satisfied for correct operation.

- Deviations Status Column Description

Term	Definition
N/S	Not In Scope
N/F	Not Fully Implemented
N/I	Not Implemented

Below table identifies the AUTOSAR requirements that are not fully implemented, implemented differently, or out of scope for the driver.

- Driver Deviations Table

Requirement	Status	Description	Notes
SWS_Fls_00145	N/S	If possible, e.g. with interrupt controlled implementations, the FLS module shall start the first round of the erase job directly within the function Fls_Erase to reduce overall runtime.	Fls driver does not support interrupt. Will be removed after ticket: AAI-902 is implemented.
SWS_Fls_00146	N/S	If possible, e.g. with interrupt controlled implementations, the FLS module shall start the first round of the write job directly within the function Fls_Write to reduce overall runtime.	Fls driver does not support interrupt. Will be removed after ticket: AAI-902 is implemented.
SWS_Fls_00232	N/S	The configuration parameter Fls↩ UseInterrupts shall switch between interrupt and polling controlled job processing if this is supported by the flash memory hardware.	Fls driver does not support interrupt. Will be removed after ticket: AAI-902 is implemented.
SWS_Fls_00233	N/S	The FLS module's implementer shall locate the interrupt service routine in Fls_Irq.c.	Fls driver does not support interrupt. Will be removed after ticket: AAI-902 is implemented.
SWS_Fls_00234	N/S	If interrupt controlled job processing is supported and enabled with the configuration parameter FlsUse↩ Interrupts, the interrupt service routine shall <continue>	Fls driver does not support interrupt. Will be removed after ticket: AAI-902 is implemented.

- As a deviation from standard: Fls_[VariantName]_PBcfg.c files will contain the definition for all parameters that are variant aware, independent of the configuration class that will be selected (PC, LT, PB) Fls_Cfg.c, Fls_Cfg.h file will contain the definition for all parameters that are not variant aware

3.5 Driver Limitations

3.5.1 For internal flash.

- With Fls_Write, both u32TargetAddress and u32Length must be double words aligned. If not, a STATUS↵_FTFC_FLS_IP_ERROR_INPUT_PARAM error code will be thrown at IP layer leading to FLS_E_W↵RITE_FAILED at HLD.

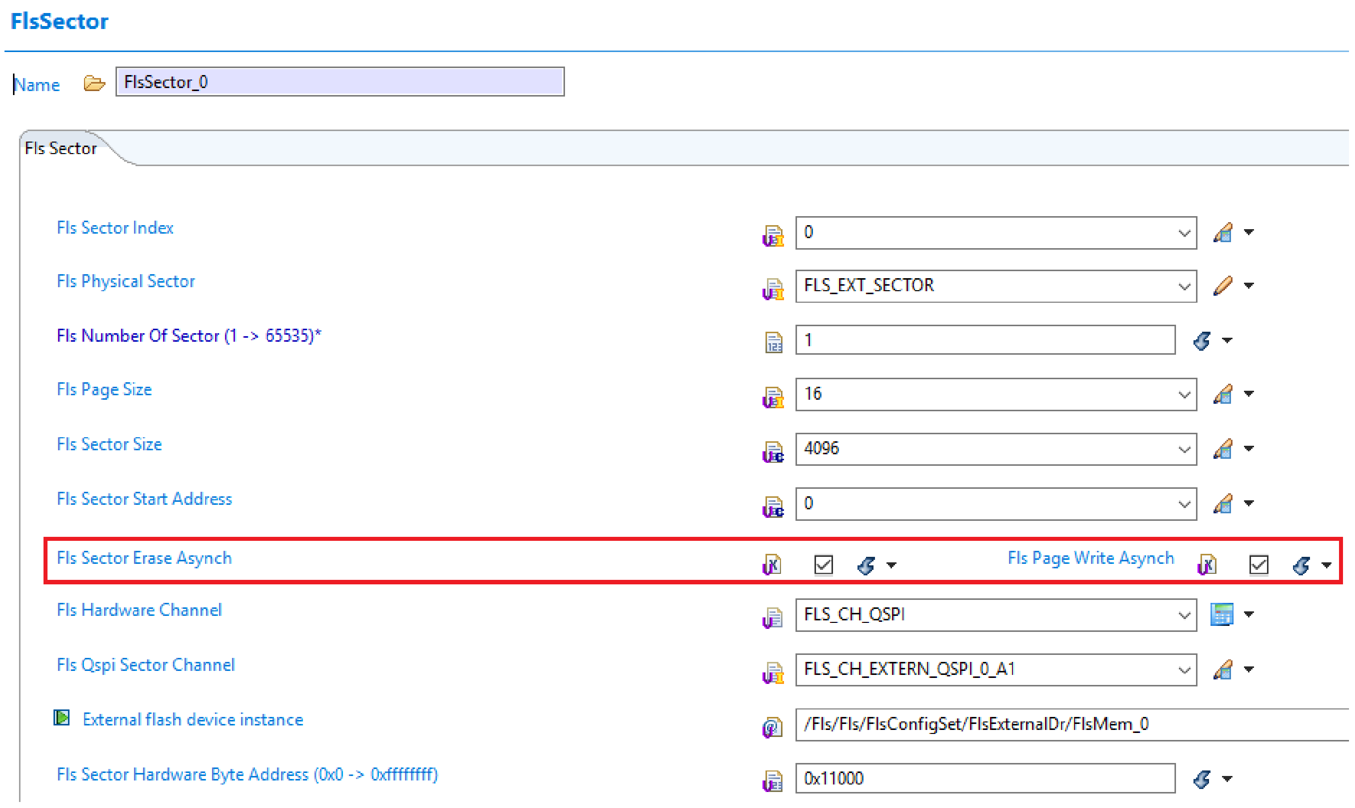
Note:

- On S32K148 derivative, PCCRMR[R2] should be programmed as 00b (NonCacheable) as S32K148 FlexNVMs region is not cacheable
- For more information, please refer to the LMEM chapter in the Reference manual and "Data Cache Restrictions" chapter in the Integration Manual
- The SFDP feature only supports standard JEDEC tables, all vendor-specific tables will be skipped

3.6 Driver usage and configuration tips

3.6.1 Introduction

For internal flash sectors, it's possible to modify the behavior of sector erase / page write using two configuration parameters (FlsSectorEraseAsynch, FlsPageWriteAsynch) in FlsSector TAB.



FlsSector

Name: FlsSector_0

Fls Sector

Fls Sector Index	0
Fls Physical Sector	FLS_EXT_SECTOR
Fls Number Of Sector (1 -> 65535)*	1
Fls Page Size	16
Fls Sector Size	4096
Fls Sector Start Address	0
Fls Sector Erase Asynch	✓
Fls Page Write Asynch	✓
Fls Hardware Channel	FLS_CH_QSPI
Fls Qspi Sector Channel	FLS_CH_EXTERN_QSPI_0_A1
External flash device instance	/Fls/Fls/FlsConfigSet/FlsExternalDr/FlsMem_0
Fls Sector Hardware Byte Address (0x0 -> 0xffffffff)	0x11000

If FlsSectorEraseAsynch/FlsPageWriteAsynch are enabled sector erase / page write job in the Fls_MainFunction are executed asynchronously, it means that Fls_MainFunction will not wait (not blocking) for completion of high voltage operation.

If FlsSectorEraseAsynch/FlsPageWriteAsynch are disabled sector erase / page write job are executed synchronously, which means sector erase / page write job are blocking and any high voltage operation will be completed during one Fls_Mainfunction.

3.6.2 Avoiding RWW problem

To avoid RWW (Read While Write) problems on the internal flash, the FLS driver provides the FlsAcLoadOn←JobStart configuration parameter. If it is set to true the Fls driver will load the flash access code routine to RAM whenever an erase or write job is started and unload (overwrite) it after that job has been finished or cancelled.

FlsAcLoadOnJobStart functionality can be used only in case of Sync Mode, in which case the flash access code is loaded to RAM and therefore the flash driver shouldn't have RWW problems; if FlsAcLoadOnJobStart is set to false the sector erased / page written must belong to flash array / partition different from flash array / partition the application is executing from.

In case of Async operations it is only possible to erase / write to flash array different from flash array the application is executing from. This mode is usable only if the platform supports different Read While Write partitions or if the entire code is executed from RAM, during the flash modify operation.

Note:

1. For internal flash, the flash driver use the sector erase / page write access code to clear the FSTAT:CCIF bit and wait for completion of high voltage operation (and therefore incompatible with Async operation).
2. The flash module is further divided into partitions/blocks that determine locations for valid read-while-write (RWW) operations(Ex: Program flash block 0 and Data flash/FlexNvm). While the embedded flash memory is performing a 'write' (program or erase) to a given partition, it can simultaneously perform a read from any other partition.
3. FlsAcCallback should be in located in RAM if FlsACLoadOnJob is true to avoid RWW problem.

3.6.3 Flash memory physical sectors unlock example

For unprotecting internal flash sectors, the flash field configuration locations corresponding to sector protection have to be erased (Addresses 0x0_0408 - 0x0_040B and 0x0_040F), reprogrammed if needed and the chip reset.

Care has to be taken when programming the flash configuration field, so that FSEC location (0x0_040C) is reprogrammed to value 0xFE after erase, and all configuration locations are erased or programmed as needed.

Code example for resetting configuration field using direct register access.

```
/* Erase flash sector containing configuration field */
FTFC->FCCOB0 = 0x09;    /* Erase sector */
FTFC->FCCOB1 = 0x00;    /* Address 0x0_0000*/
FTFC->FCCOB2 = 0x00;    /* Address */
FTFC->FCCOB3 = 0x00;    /* Address */

/* Program flash configuration field */
FTFC->FCCOB0 = 0x07;    /* Program phrase */
FTFC->FCCOB1 = 0x08;    /* Address 0x0_0408*/
FTFC->FCCOB2 = 0x04;    /* Address */
FTFC->FCCOB3 = 0x00;    /* Address */
FTFC->FCCOB4 = 0xFF;    /* Data for flash location 0x0_040B, FPROT3 */
FTFC->FCCOB5 = 0xFF;    /* Data for flash location 0x0_040A, FPROT2 */
FTFC->FCCOB6 = 0xFF;    /* Data for flash location 0x0_0409, FPROT1 */
FTFC->FCCOB7 = 0xFF;    /* Data for flash location 0x0_0408, FPROT0 */
FTFC->FCCOB8 = 0xFF;    /* Data for flash location 0x0_040F, FDPROT */
FTFC->FCCOB9 = 0xFF;    /* Data for flash location 0x0_040E, FEPROT */
FTFC->FCCOBA = 0xFF;    /* Data for flash location 0x0_040D, FOPT */
FTFC->FCCOBB = 0xFE;    /* Data for flash location 0x0_040C, FSEC */

/* Reset */
```

3.6.4 Development Error Description

Error Code	Value	Condition triggering the error
FLS_E_PARAM_CONFIG	1	API service called with wrong parameter
FLS_E_PARAM_ADDRESS	2	u32TargetAddress is not in range and aligned to first byte of flash sector
FLS_E_PARAM_LENGTH	3	u32TargetAddress is not in range and aligned to last byte of flash sector
FLS_E_PARAM_DATA	4	API service called with wrong parameter
FLS_E_UNINIT	5	API service called without module initialization
FLS_E_BUSY	6	API service called while driver still busy
FLS_E_PARAM_POINTER	10	API service called with NULL_PTR passed

3.7 Runtime errors

- The driver supports runtime generation of the errors listed in the table:

Error code	Function	Condition triggering the error
FLS_E_VERIFY_ERASE_FAILED	Fls_MainFunction()	Verify erasing operation failed before writing a flash block
		Verify erasing operation failed after erasing a flash block

Error code	Function	Condition triggering the error
FLS_E_VERIFY_WRITE_FAILED	Fls_MainFunction()	Verify writing operation failed after writing a flash block
FLS_E_TIMEOUT	Fls_Init()	Access timeout value to the flash controller has been exceeded
	Fls_MainFunction()	Maximum read / write / compare / erase time has been exceeded

- The driver supports Transient faults generation of the errors listed in the table:

Error Code	Function	Condition triggering the error
FLS_E_ERASE_FAILED	Fls_MainFunction()	A flash erase job fails due to a hardware error
FLS_E_WRITE_FAILED		A flash write job fails due to a hardware error
FLS_E_READ_FAILED		A flash read job fails due to a hardware error
FLS_E_COMPARE_FAILED		A flash compare job fails due to a hardware error
FLS_E_UNEXPECTED_FLASH_ID	Fls_Init()	The hardware ID of the external flash device mismatched the corresponding configuration parameter

- Development Error Description

Error Code	Value	Condition triggering the error
FLS_E_PARAM_CONFIG	1	API service called with wrong parameter
FLS_E_PARAM_ADDRESS	2	u32TargetAddress is not in range and aligned to first byte of flash sector
FLS_E_PARAM_LENGTH	3	u32TargetAddress is not in range and aligned to last byte of flash sector
FLS_E_PARAM_DATA	4	NULL_PTR == SourceAddressPtr
FLS_E_UNINIT	5	API service called without module initialization
FLS_E_BUSY	6	PI service called while driver still busy
FLS_E_PARAM_POINTER	10	NULL_PTR passed

3.8 Symbolic Names Disclaimer

All containers having symbolicNameValue set to TRUE in the AUTOSAR schema will generate defines like:

```
#define <Mip>Conf_<Container_ShortName>_<Container_ID>
```

For this reason it is forbidden to duplicate the names of such containers across the RTD configurations or to use names that may trigger other compile issues (e.g. match existing `#ifdefs` arguments).

Chapter 4

Tresos Configuration Plug-in

This chapter describes the Tresos configuration plug-in for the driver. All the parameters are described below.

- Module Fls
  - Container FlsConfigSet
    - Parameter FlsAcErase
    - Parameter FlsAcWrite
    - Parameter FlsAcErasePointer
    - Parameter FlsAcWritePointer
    - Parameter FlsCallCycle
    - Parameter FlsDefaultMode
    - Parameter FlsACCallback
    - Parameter FlsJobEndNotification
    - Parameter FlsJobErrorNotification
    - Parameter FlsStartFlashAccessNotif
    - Parameter FlsFinishedFlashAccessNotif
    - Parameter FlsReadFunctionCallout
    - Parameter FlsQspiInitCallout
    - Parameter FlsQspiResetCallout
    - Parameter FlsQspiErrorCheckCallout
    - Parameter FlsQspiEccCheckCallout
    - Parameter FlsMaxReadFastMode
    - Parameter FlsMaxReadNormalMode
    - Parameter FlsMaxWriteFastMode
    - Parameter FlsMaxWriteNormalMode
    - Parameter FlsProtection
    - Container FlsExternalDriver
      - Reference FlsSpiReference
      - Container ControllerCfg
      - Parameter FlsHwUnitReadMode
      - Parameter FlsSerialFlashA1Size
      - Parameter FlsSerialFlashA2Size
      - Parameter FlsSerialFlashB1Size

- Parameter FlsSerialFlashB2Size
- Parameter FlsHwUnitSamplingModeA
- Parameter FlsHwUnitSamplingModeB
- Parameter IdleSignalDriveIOFB3HighLvl
- Parameter IdleSignalDriveIOFB2HighLvl
- Parameter IdleSignalDriveIOFA3HighLvl
- Parameter IdleSignalDriveIOFA2HighLvl
- Parameter FlsHwUnitSamplingEdge
- Parameter FlsHwUnitSamplingDly
- Parameter FlsHwUnitDqsLatencyEnable
- Parameter FlsHwUnitTdh
- Parameter FlsHwUnitTcsh
- Parameter FlsHwUnitTcss
- Parameter FlsHwUnitColumnAddressWidth
- Parameter FlsHwUnitWordAddressable
- Container FlsAhbBuffer
- Parameter FlsAhbBufferInstance
- Parameter FlsAhbBufferMasterId
- Parameter FlsAhbBufferSize
- Parameter FlsAhbBufferAllMasters
- Container FlsHwUnitSpecificSettings
- Parameter FlsHwUnitInputClockSelelect
- Parameter FlsHwUnitInternalRefClockSelect
- Parameter FlsHwUnitProgrammableDivider
- Parameter FlsHwUnitInvertClkDqsA
- Parameter FlsHwUnitInvertClkDqsB
- Parameter FlsHwUnitFineDelayA
- Parameter FlsHwUnitFineDelayB
- Container MemCfg
- Parameter MemCfgSize
- Parameter MemCfgPageSize
- Reference MemCfgReadLUT
- Reference MemCfgWriteLUT
- Reference MemCfgRead0xxLUT
- Reference MemCfgRead0xxLUTAHB
- Reference ctrlAutoCfgPtr
- Container MemCfgReadIdSettings
- Parameter MemCfgReadIdSize
- Parameter FlsQspiDeviceId
- Reference MemCfgReadIdLUT
- Container MemCfgEraseSettings
- Parameter MemCfgErase1Size
- Parameter MemCfgErase2Size
- Parameter MemCfgErase3Size
- Parameter MemCfgErase4Size
- Reference MemCfgErase1LUT
- Reference MemCfgErase2LUT

- Reference MemCfgErase3LUT
- Reference MemCfgErase4LUT
- Reference ChipEraseLUT
- Container statusConfig
- Parameter regSize
- Parameter busyOffset
- Parameter busyValue
- Parameter writeEnableOffset
- Parameter blockProtectionOffset
- Parameter blockProtectionWidth
- Parameter blockProtectionValue
- Reference statusRegInitReadLut
- Reference statusRegReadLut
- Reference statusRegWriteLut
- Reference writeEnableSRLut
- Reference writeEnableLut
- Container suspendSettings
- Reference eraseSuspendLut
- Reference eraseResumeLut
- Reference programSuspendLut
- Reference programResumeLut
- Container resetSettings
- Parameter resetCmdCount
- Reference resetCmdLut
- Container initResetSettings
- Parameter resetCmdCount
- Reference resetCmdLut
- Container initConfiguration
- Parameter opType
- Parameter addr
- Parameter size
- Parameter shift
- Parameter width
- Parameter value
- Reference command1Lut
- Reference command2Lut
- Reference weLut
- Reference ctrlCfgPtr
- Container FlsLUT
- Parameter FlsLUTIndex
- Container FlsInstructionOperandPair
- Parameter FlsInstrOperPairIndex
- Parameter FlsLUTInstruction
- Parameter FlsLUTPad
- Parameter FlsLUTOperand
- Container FlsController
- Parameter ControllerName

    * Parameter FlsUseInterrupts
    * Parameter FlsVersionInfoApi
    * Parameter FlsECCCheck
    * Parameter FlsECCHandlingProtectionHook
    * Parameter FlsEraseVerificationEnabled
    * Parameter FlsWriteVerificationEnabled
    * Parameter FlsMaxEraseBlankCheck
    * Parameter FlsTimeoutSupervisionEnabled
    * Parameter FlsTimeoutMethod
    * Parameter FlsAsyncWriteTimeout
    * Parameter FlsAsyncEraseTimeout
    * Parameter FlsSyncWriteTimeout
    * Parameter FlsSyncEraseTimeout
    * Parameter FlsAbortTimeout
    * Parameter FlsQspiIpTimeoutOsifCounterType
    * Parameter FlsQspiSyncReadTimeout
    * Parameter FlsQspiAsyncWriteTimeout
    * Parameter FlsQspiAsyncEraseTimeout
    * Parameter FlsQspiSyncWriteTimeout
    * Parameter FlsQspiSyncEraseTimeout
    * Parameter FlsQspiCommandCompleteTimeout
    * Parameter FlsQspiResetTimeout
    * Parameter FlsQspiFlashInitTimeout
    * Parameter FlsQspiSoftwareResetDelay
    * Parameter FlsQspiTxBufferResetDelay
    * Parameter FlsQspiWriteEnableRetries
    * Reference FlsEcucPartitionRef
- Container FlsPublishedInformation
    * Parameter FlsAcLocationErase
    * Parameter FlsAcLocationWrite
    * Parameter FlsAcSizeErase
    * Parameter FlsAcSizeWrite
    * Parameter FlsEraseTime
    * Parameter FlsErasedValue
    * Parameter FlsECCValue
    * Parameter FlsExpectedHwId
    * Parameter FlsSpecifiedEraseCycles
    * Parameter FlsWriteTime
- Container CommonPublishedInformation
    * Parameter ArReleaseMajorVersion
    * Parameter ArReleaseMinorVersion

  * ∗ Parameter ArReleaseRevisionVersion
  * ∗ Parameter ModuleId
  * ∗ Parameter SwMajorVersion
  * ∗ Parameter SwMinorVersion
  * ∗ Parameter SwPatchVersion
  * ∗ Parameter VendorApiInfix
  * ∗ Parameter VendorId

4.1 Module Fls

Configuration of the Fls (internal or external flash driver) module.

Included containers:

- FlsConfigSet
- AutosarExt
- FlsGeneral
- FlsPublishedInformation
- CommonPublishedInformation

Property	Value
type	ECUC-MODULE-DEF
lowerMultiplicity	1
upperMultiplicity	Infinite
postBuildVariantSupport	true
supportedConfigVariants	VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

4.2 Container FlsConfigSet

Container for runtime configuration parameters of the flash driver.

Implementation Type: Fls_ConfigType.

Included subcontainers:

- FlsExternalDriver
- FlsSectorList

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.3 Parameter FlsAcErase

Address offset in RAM to which the erase flash access code shall be loaded.

Used as function pointer to access the erase flash access code.

Note: To use Fls Access Code Erase be sure Fls Access Code Erase Pointer is NULL or NULL_PTR.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1073751296
max	4294967295
min	0

4.4 Parameter FlsAcWrite

Address offset in RAM to which the write flash access code shall be loaded.

Used as function pointer to access the write flash access code.

Note: To use Fls Access Code Write be sure Fls Access Code Write Pointer is NULL or NULL_PTR.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC

Property	Value
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1073751296
max	4294967295
min	0

4.5 Parameter FlsAcErasePointer

Vendor specific: Pointer in RAM to which the erase flash access code shall be loaded.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	NULL_PTR

4.6 Parameter FlsAcWritePointer

Vendor specific: Pointer in RAM to which the write flash access code shall be loaded.

Used as function pointer to access the write flash access code.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	NULL_PTR

4.7 Parameter FlsCallCycle

Cycle time of calls of the flash driver main function

Property	Value
type	ECUC-FLOAT-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0.2
max	1.0
min	0.0

4.8 Parameter FlsDefaultMode

This parameter is the default FLS device mode after initialization.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

Property	Value
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	MEMIF_MODE_SLOW
literals	['MEMIF_MODE_FAST', 'MEMIF_MODE_SLOW']

4.9 Parameter FlsACCallback

Vendor specific: Mapped to the Access Code Callback provided by some upper layer module, typically the Wdg module.

Note: Disable the Access Code Callback to have it set as NULL_PTR.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	Fls_AC_Callback

4.10 Parameter FlsJobEndNotification

Mapped to the job end notification routine provided by some upper layer module, typically the Fee module.

Note: Disable the end notification to have it set as NULL_PTR

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true

Property	Value
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	Fee_JobEndNotification

4.11 Parameter FlsJobErrorNotification

Mapped to the job error notification routine provided by some upper layer module, typically the Fee module.

Note: Disable the error notification to have it set as NULL_PTR

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	Fee_JobErrorNotification

4.12 Parameter FlsStartFlashAccessNotif

Vendor specific: Start flash access. If configured, this notification will be called before any flash memory access.

It is called before flash memory read accesses(in read, compare, verify write, verify erase jobs) and

before flash memory program operations(in write and erase jobs).

The purpose of this notification together with FlsFinishedFlashAccess, is to ensure that, if needed, no other

executed code(other tasks, cores, masters) will access the affected flash area simultaneously with the access

initiated by the driver. For more details, see Integration manual, chapter 5. Module requirements.

Note: Disable the error notification to have it set as NULL_PTR

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	Fls_StartFlashAccessNotif

4.13 Parameter FlsFinishedFlashAccessNotif

Vendor specific: Finished flash access. If configured, this notification will be called after any flash memory access.

It is called after flash memory read accesses(in read, compare, verify write, verify erase jobs).

The purpose of this notification together with FlsStartFlashAccess, is to ensure that, if needed, no other

executed code(other tasks, cores, masters) will access the affected flash area simultaneously with the access

initiated by the driver. For more details, see Integration manual, chapter 5. Module requirements.

Note: Disable the error notification to have it set as NULL_PTR

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	Fls_FinishedFlashAccessNotif

4.14 Parameter FlsReadFunctionCallout

Vendor specific: The callout for the user to check for ECC errors for Internal Flash memories.

In this callout, the user can schedule a task that reads from flash memory

to a read source buffer and check/handle for an ECC exception.

If an exception occurs, a descriptor regarding the faulty line number

that caused the ECC and the state of the task should be updated.

Note: Inside a task, the flow is not endangered in case of an ECC exception, as the task can be forcibly terminated in that case.

(please see the chapter 'ECC Management on Flash' in IM for more information)

- Disable: Read and Compare functions will be handled by driver

- Enable: Read and Compare functions will be handled by users.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	FlsReadFunctionCallout

4.15 Parameter FlsQspiInitCallout

Vendor specific: Callout function called by the driver at the end of the QSPI Init phase.

The intended purpose of this callout is to provide to the application the

possibility of performing additional configuration to the QSPI hardware IP or

to the external memories connected(for ex: sending the lock/unlock sequences

for the external flash sectors, altering QSPI IP timing, etc.)

Note: Disable the callout in order to have it set as NULL_PTR.

Note: The callout can be configured only if the FlsExternalDriver is enabled.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	FlsQspiInitCallout

4.16 Parameter FlsQspiResetCallout

Vendor specific: Callout function called by the driver at the beginning of a new job. The intended purpose of this callout is to provide to the application thepossibility of reseting the external memory to an idle and error free state.

If the callout is disabled, at the beginnig of a new job the Fls_MainFunction will check the external memory status and if not, poll and wait for it to become idle.

If the callout is enabled and the memory is not idle, the Fls_MainFunction will also call the configured function to allow the application to send extra commands to the external memory(software reset, abort any suspended operation, error flags clearing, etc.)

Note: Disable the callout in order to have it set as NULL_PTR.

Note: The callout can be configured only if the FlsExternalDriver is enabled.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	FlsQspiResetCallout

4.17 Parameter FlsQspiErrorCheckCallout

Vendor specific: Callout function called by the driver at the end of each program and erase job

The intended purpose of this callout is to provide to the application the

possibility of interrogating the error status of the memory after each program and erase job.

The application should check any error or status bits available and reset the memory after interrogation

in case an error condition was detected.

If the callout is enabled, at the end of each job, the callout is called and the return

value is checked to determine if there was any error during the memory operation.

Return values: E_OK(0) E_NOT_OK(1).

If E_OK(0) is received, the job is considered successful.

If E_NOT_OK(1) is received, the job is considered unsuccessful and marked as failed.

If the callout is disabled, the job is assumed as successful from the memory status point of view.

Note: Disable the callout in order to have it set as NULL_PTR.

Note: The callout can be configured only if the FlsExternalDriver is enabled.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	FlsQspiErrorCheckCallout

4.18 Parameter FlsQspiEccCheckCallout

Vendor specific: Callout function called by the driver at the end of each read operation

The intended purpose of this callout is to provide to the application the possibility of interrogating the ECC status of the memory after each read operation.

The callout provide the hardware channel, start address and the size of the read operation.

The application should check there is any ECC error in the current read data

If the callout is enabled, at the end of each read operation, the callout is called and the return

value is checked to determine if there was any error during the memory operation.

Return values: E_OK(0) E_NOT_OK(1).

If E_OK(0) is received, the job is considered successful.

If E_NOT_OK(1) is received, the job is considered unsuccessful and marked as failed.

If the callout is disabled, the job is assumed as successful from the memory status point of view.

Note: Disable the callout in order to have it set as NULL_PTR.

Note: The callout can be configured only if the FlsExternalDriver is enabled.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	FlsQspiEccCheckCallout

4.19 Parameter FlsMaxReadFastMode

The maximum number of bytes to read or compare in one cycle of the flash driver's job processing function in fast mode.

Note: If external sectors are configured and if FlsHwUnitWordAddressable is set,

the FlsMaxReadFastMode must be an even value(two bytes aligned).

Property	Value
type	ECUC-INTEGER-PARAM-DEF

Property	Value
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1048576
max	4294967295
min	0

4.20 Parameter FlsMaxReadNormalMode

The maximum number of bytes to read or compare in one cycle of the flash driver's job processing function in normal mode.

Note: If external sectors are configured and if FlsHwUnitWordAddressable is set,

the FlsMaxReadNormalMode must be an even value(two bytes aligned).

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1024
max	4294967295
min	0

4.21 Parameter FlsMaxWriteFastMode

The maximum number of bytes to write in one cycle of the flash driver's job processing function in fast mode.

Note: If external sectors are configured, the FlsMaxWriteFastMode must be an integer multiple of the FlsPageSize.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	256
max	4294967295
min	0

4.22 Parameter FlsMaxWriteNormalMode

The maximum number of bytes to write in one cycle of the flash driver's job processing function in normal mode.

Note: If external sectors are configured, the FlsMaxWriteFastMode must be an integer multiple of the FlsPageSize.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	8
max	4294967295
min	0

4.23 Parameter FlsProtection

Erase/write protection settings.Note:Not supported by the driver.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.24 Container FlsExternalDriver

This container is present for external Flash drivers only. Internal Flash drivers do not use the parameter listed in this container, hence its multiplicity is 0 for internal drivers.

Included subcontainers:

- ControllerCfg
- MemCfg
- FlsController
- FlsMem

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD

4.25 Reference FlsSpiReference

Reference to SPI sequence. Not used in current implementation.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	true
destination	/AUTOSAR/EcucDefs/Spi/SpiDriver/SpiSequence

4.26 Container ControllerCfg

Vendor specific: Container for the configuration of the avaialable external flash memory hardware units.

Included subcontainers:

- FlsAhbBuffer
- FlsHwUnitSpecificSettings

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE

4.27 Parameter FlsHwUnitReadMode

Vendor specific: The hardware unit read mode:

QSPI_IP_DATA_RATE_SDR (single data rate) which samples incoming data on a single edge.

QSPI_IP_DATA_RATE_DDR (double data rate) which samples incoming data on both edges.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF

Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_DATA_RATE_SDR
literals	['QSPI_IP_DATA_RATE_SDR', 'QSPI_IP_DATA_RATE_DDR']

4.28 Parameter FlsSerialFlashA1Size

Vendor specific: Size of flash device connected to side A1 of the controller. Set to 0 if no flash device is connected.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.29 Parameter FlsSerialFlashA2Size

Vendor specific: Size of flash device connected to side A2 of the controller. Set to 0 if no flash device is connected.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.30 Parameter FlsSerialFlashB1Size

Vendor specific: Size of flash device connected to side B1 of the controller. Set to 0 if no flash device is connected.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.31 Parameter FlsSerialFlashB2Size

Vendor specific: Size of flash device connected to side B2 of the controller. Set to 0 if no flash device is connected.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.32 Parameter FlsHwUnitSamplingModeA

Vendor specific: It selects DQS clock for sampling read data at Flash A QuadSPI port:

QSPI_IP_READ_MODE_INTERNAL_DQS = DQS internal (Default).

QSPI_IP_READ_MODE_LOOPBACK = Pad loopback.

QSPI_IP_READ_MODE_LOOPBACK_DQS = DQS pad loopback.

QSPI_IP_READ_MODE_EXTERNAL_DQS = External DQS.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_READ_MODE_INTERNAL_DQS
literals	['QSPI_IP_READ_MODE_INTERNAL_DQS', 'QSPI_IP_READ_MODE↩ _LOOPBACK']

4.33 Parameter FlsHwUnitSamplingModeB

Vendor specific: It selects DQS clock for sampling read data at Flash B QuadSPI port:

QSPI_IP_READ_MODE_INTERNAL_DQS = DQS internal (Default).

QSPI_IP_READ_MODE_LOOPBACK = Pad loopback.

QSPI_IP_READ_MODE_LOOPBACK_DQS = DQS pad loopback.

QSPI_IP_READ_MODE_EXTERNAL_DQS = External DQS.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_READ_MODE_INTERNAL_DQS
literals	['QSPI_IP_READ_MODE_INTERNAL_DQS', 'QSPI_IP_READ_MODE↩ _LOOPBACK', 'QSPI_IP_READ_MODE_EXTERNAL_DQS']

4.34 Parameter IdleSignalDriveIOFB3HighLvl

Vendor specific: Idle Signal Drive IOFB[3] Flash B. This bit determines the logic level the IOFB[3] output of the QuadSPI module is driven to in the inactive state.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.35 Parameter IdleSignalDriveIOFB2HighLvl

Vendor specific: Idle Signal Drive IOFB[2] Flash B. This bit determines the logic level the IOFB[2] output of the QuadSPI module is driven to in the inactive state.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.36 Parameter IdleSignalDriveIOFA3HighLvl

Vendor specific: Idle Signal Drive IOFA[3] Flash A. This bit determines the logic level the IOFA[3] output of the QuadSPI module is driven to in the inactive state.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.37 Parameter IdleSignalDriveIOFA2HighLvl

Vendor specific: Idle Signal Drive IOFA[2] Flash A. This bit determines the logic level the IOFA[2] output of the

QuadSPI module is driven to in the inactive state.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.38 Parameter FlsHwUnitSamplingEdge

Vendor specific: Full-speed phase selection for SDR instructions.

This field selects the edge of the sampling clock valid for full-speed commands.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_SAMPLE_PHASE_NON_INVERTED
literals	['QSPI_IP_SAMPLE_PHASE_NON_INVERTED', 'QSPI_IP_SAMPLE_↩ PHASE_INVERTED']

4.39 Parameter FlsHwUnitSamplingDly

Vendor specific: Full-speed delay selection for internal/pad loop back DQS sampling.

This field selects the delay in accordance with the reference edge for the valid sample point.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_SAMPLE_DELAY_SAME_DQS
literals	['QSPI_IP_SAMPLE_DELAY_SAME_DQS', 'QSPI_IP_SAMPLE_DELA↩ Y_HALFCYCLE_EARLY_DQS']

4.40 Parameter FlsHwUnitDqsLatencyEnable

Vendor specific: DQS Latency Enable. Feature used to support external devices which add latency cycles

in the DQS signal. When no signal is provided by the external devices, data is not sampled,

thus the memory stretches the timing.

DQS Latency is applicable only for DQS sampling mode: FLS_EXTERNAL_DQS.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.41 Parameter FlsHwUnitTdh

Vendor specific: TDH: Serial flash data in hold time. Should be set to QSPI_IP_FLASH_DATA_ALIGN_REFCLK for QSPI_IP_DATA_RATE_SDR mode.

TDH parameter delays data sent to flash, in order to meet the input hold time requirement of flash.

QSPI_IP_FLASH_DATA_ALIGN_REFCLK = Data aligned with the posedge of Internal reference clock of QuadSPI.

QSPI_IP_FLASH_DATA_ALIGN_2X_REFCLK = Data aligned with 2x serial flash half clock.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_FLASH_DATA_ALIGN_REFCLK
literals	['QSPI_IP_FLASH_DATA_ALIGN_REFCLK', 'QSPI_IP_FLASH_DATA_ALIGN_2X_REFCLK']

4.42 Parameter FlsHwUnitTcsh

Vendor specific: TCSH: Serial flash CS hold time in terms of serial flash clock cycles.

A bigger value will release the CS signal later after the transaction ends.

The actual delay between chip select and clock is defined as:

TCSH = 1 SCK clk if N= 0/1 else, N SCK clk if N>1, where N is the setting of TCSH

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	3
max	15
min	0

4.43 Parameter FlsHwUnitTcss

Vendor specific: TCSS: Serial flash CS setup time in terms of serial flash clock cycles.

A bigger value will pull the CS signal earlier before the transaction starts.

The actual delay between chip select and clock is defined as:

TCSS = 0.5 SCK clk if N= 0/1 else, N+0.5 SCK clk if N>1, where N is the setting of TCSS.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	3
max	15
min	0

4.44 Parameter FlsHwUnitColumnAddressWidth

Vendor specific: Column Address Space. Defines the width of the column address.

Example: If the coulmn address is for example [2:0] of QSPI_SFAR/AHB address,

then CAS must be 3. If there is no column address separation in any

serial flash this value must be programmed to 0.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true

Property	Value
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	15
min	0

4.45 Parameter FlsHwUnitWordAddressable

Vendor specific: Defines whether the serial flash is a byte addressable flash or a word addressable flash.

According to this bit configuration the address is re-mapped to the flash interface.

DISABLED: Byte addressable serial flash mode.

ENABLED: Word (2 byte) addressable serial flash mode. If the

incoming address is 0x2004, the controller re-maps this address

to access the flash location 0x1002.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.46 Container FlsAhbBuffer

Container for the configuration of the AHB read buffers. Holds the configuration for each

AHB buffer configured for AHB read mode.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	4
upperMultiplicity	4
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.47 Parameter FlsAhbBufferInstance

Vendor specific: Selects the AHB buffer instance for which this configuration applies.

If an instance is not present, the corresponding AHB buffer will be configured with size 0.

The size of the AHB_BUFFER_3 instance will be configured to at least the selected size, or more, up until the maximum

value is reached. For more details about the maximum avaialable size see chip specific details.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	AHB_BUFFER_1
literals	['AHB_BUFFER_0', 'AHB_BUFFER_1', 'AHB_BUFFER_2', 'AHB_BUF↩ FER_3']

4.48 Parameter FlsAhbBufferMasterId

Vendor specific: The ID of the AHB master associated with this buffer. Any AHB access with this master port

number is routed to this buffer. It must be ensured that the master IDs associated with all

buffers must be different.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	15
min	0

4.49 Parameter FlsAhbBufferSize

Vendor specific: The size allocated to this AHB Buffer instance. The minimum size is 8 bytes, the maximum size is the entire AHB Buffer.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.50 Parameter FlsAhbBufferAllMasters

Vendor specific: When set, buffer3 acts as an all-master buffer. Any AHB access with a master port number not matching with the master ID of buffer0 or buffer1 or buffer2 is routed to buffer3.
When set, the Master ID parameter for this buffer is ignored.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.51 Container FlsHwUnitSpecificSettings

Vendor specific: Container for clock options and chip settings.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.52 Parameter FlsHwUnitInputClockSelelect

Vendor specific: Select source for AHB read interface clock, module clock and bus interface clock.

SCLKCFG[6] bit.

Option 1: QSPI_IP_CLK_SRC_BUS_CLK.

Option 2: QSPI_IP_CLK_SRC_SYS_CLK.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF

Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_CLK_SRC_SYS_CLK
literals	['QSPI_IP_CLK_SRC_SYS_CLK', 'QSPI_IP_CLK_SRC_BUS_CLK']

4.53 Parameter FlsHwUnitInternalRefClockSelect

Vendor specific: Internal reference clock (async clock domain) source selection for Quadspi.

SCLKCFG[4] bit.

Option 1: QSPI_IP_CLK_REF_PLL_DIV1.

Option 2: QSPI_IP_CLK_REF_FIRC_DIV1.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_CLK_REF_FIRC_DIV1
literals	['QSPI_IP_CLK_REF_PLL_DIV1', 'QSPI_IP_CLK_REF_FIRC_DIV1']

4.54 Parameter FlsHwUnitProgrammableDivider

Vendor specific: Programmable divider configuration selection. Based on this devider value, the external memory clock and internal sampling clock is derived.

SOCCFG[31:29] bits.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	2
max	8
min	1

4.55 Parameter FlsHwUnitInvertClkDqsA

Vendor specific: Reference clock selection for DQS for Flash-A.

SCLKCFG[3] bit.

True : Inverted Clock from 'Fls Hw Unit Sampling Mode A' selected as DQS.

False: Clock from 'Fls Hw Unit Sampling Mode A selected as DQS.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.56 Parameter FlsHwUnitInvertClkDqsB

Vendor specific: Reference clock selection for DQS for Flash-B.

SCLKCFG[3] bit.

True : Inverted Clock from 'Fls Hw Unit Sampling Mode B' selected as DQS.

False: Clock from 'Fls Hw Unit Sampling Mode B' selected as DQS.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.57 Parameter FlsHwUnitFineDelayA

Vendor specific: Fine delay chain configuration for Flash A.

SOCCFG[7:0] bits.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	127
min	0

4.58 Parameter FlsHwUnitFineDelayB

Vendor specific: Fine delay chain configuration for Flash B.

SOCCFG[15:8] bits.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	127
min	0

4.59 Container MemCfg

Vendor specific: Container for the configuration of the avaialable external flash memory hardware units.

Included subcontainers:

- MemCfgReadIdSettings
- MemCfgEraseSettings
- statusConfig
- suspendSettings
- resetSettings
- initResetSettings
- initConfiguration
- FlsLUT

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE

4.60 Parameter MemCfgSize

Vendor specific: The size in bytes of this flash device.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.61 Parameter MemCfgPageSize

Vendor specific: The page size in bytes of this flash device.

Page size is the maximum amount of data that the flash device can write in a single write operation.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.62 Reference MemCfgReadLUT

Vendor specific: Reference to the LUT Sequence ID which will be used for read operations

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.63 Reference MemCfgWriteLUT

Vendor specific: Reference to the LUT Sequence ID which will be used for write operations

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.64 Reference MemCfgRead0xxLUT

Vendor specific: Reference to the LUT Sequence ID which will be used for optimized read operations, if supported by the device.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.65 Reference MemCfgRead0xxLUTAHB

Vendor specific: Reference to the LUT Sequence ID which will be used for optimized read operations through AHB reads, if supported by the device.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.66 Reference ctrlAutoCfgPtr

Vendor specific: Reference to configuration which will be used for initializing the controller when the flash device is initialized.

This is needed for devices which need to change controller configuration during device initialization (e.g. switch to External DQS after activating DOPI mode).

Resetting the flash device will re-apply this configuration.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/ControllerCfg

4.67 Container MemCfgReadIdSettings

Vendor specific: Container for Read Device/Manufacturer ID command

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.68 Parameter MemCfgReadIdSize

Vendor specific: The size in bytes of the information returned by the readId command.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	4
min	1

4.69 Parameter FlsQspiDeviceId

Vendor specific: External memory ID. If the associated "FLS_E_UNEXPECTED_FLASH_ID" error is enabled, at Init,

the configured value is checked against the value read from memory.

The memory ID is read from the memory using the configured READ_ID LUT sequence.

Example for a Macronix device:

Configured value of FlsQspiDeviceId = 0x3A81C2, meaning Memory density: 0x3A, Memory type: 0x81, Manufacturer ID: 0xC2.

The configured READ_ID LUT sequence schedules a read id command (ex: RDID 0x9F) with read length 3 bytes.

Note: This parameter can be configured only when Read Id LUT index reference is used.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.70 Reference MemCfgReadIdLUT

Vendor specific: Reference to the LUT Sequence ID which will be used for reading device/manufacturer Id.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.71 Container MemCfgEraseSettings

Vendor specific: Container for erase commands supported by the device

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.72 Parameter MemCfgErase1Size

Vendor specific: The size in bytes of the erased area: 2 ^ size; e.g. 0x0C means 4 Kbytes

Property	Value
type	ECUC-INTEGER-PARAM-DEF

Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	12
max	32
min	1

4.73 Parameter MemCfgErase2Size

Vendor specific: The size in bytes of the erased area: 2 ^ size; e.g. 0x0C means 4 Kbytes

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	12
max	32
min	1

4.74 Parameter MemCfgErase3Size

Vendor specific: The size in bytes of the erased area: 2 ^ size; e.g. 0x0C means 4 Kbytes

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	12
max	32
min	1

4.75 Parameter MemCfgErase4Size

Vendor specific: The size in bytes of the erased area: 2 ^ size; e.g. 0x0C means 4 Kbytes

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	12
max	32
min	1

4.76 Reference MemCfgErase1LUT

Vendor specific: Reference to the LUT Sequence ID for erase type 1.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.77 Reference MemCfgErase2LUT

Vendor specific: Reference to the LUT Sequence ID for erase type 2.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.78 Reference MemCfgErase3LUT

Vendor specific: Reference to the LUT Sequence ID for erase type 3.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD

Property	Value
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.79 Reference MemCfgErase4LUT

Vendor specific: Reference to the LUT Sequence ID for erase type 4.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.80 Reference ChipEraseLUT

Vendor specific: Reference to the LUT Sequence ID for chip erase command.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.81 Container statusConfig

Vendor specific: Container for settings related to the status register of the flash device

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.82 Parameter regSize

Vendor specific: The size in bytes of the status register

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	4
min	1

4.83 Parameter busyOffset

Vendor specific: Position of "busy" bit inside status register. This bit is indicates whether the device is busy with a high voltage operation or not.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	31
min	0

4.84 Parameter busyValue

Vendor specific: Value of "busy" bit which indicates that the device is busy; can be 0 or 1

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	1
min	0

4.85 Parameter writeEnableOffset

Vendor specific: Position of "write enable" bit inside the status register

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP

Property	Value
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	31
min	0

4.86 Parameter blockProtectionOffset

Vendor specific: Offset of block protection bits inside the status register

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	2
max	31
min	0

4.87 Parameter blockProtectionWidth

Vendor specific: Width of block protection bitfield inside the status register

A value of 0 disables protection setting.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP

Property	Value
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	4
max	32
min	0

4.88 Parameter blockProtectionValue

Vendor specific: Value of block protection bitfield inside the status register

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	15
min	0

4.89 Reference statusRegInitReadLut

Vendor specific: Reference to the LUT Sequence ID for Read status register command.

This sequence is used during the initializaton stage.

For example if the initial state of the flash is SPI, this should be a SPI sequence.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.90 Reference statusRegReadLut

Vendor specific: Reference to the LUT Sequence ID for Read status register command.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.91 Reference statusRegWriteLut

Vendor specific: Reference to the LUT Sequence ID for Write status register command.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false

Property	Value
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.92 Reference writeEnableSRLut

Vendor specific: Reference to the LUT Sequence ID for Status register write enable command.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.93 Reference writeEnableLut

Vendor specific: Reference to the LUT Sequence ID for Write enable command.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE

Property	Value
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.94 Container suspendSettings

Vendor specific: Container related to write/erase suspend and resume commands, if supported by the device.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.95 Reference eraseSuspendLut

Vendor specific: Reference to the LUT Sequence ID for Erase suspend command.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.96 Reference eraseResumeLut

Vendor specific: Reference to the LUT Sequence ID for Erase resume command.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.97 Reference programSuspendLut

Vendor specific: Reference to the LUT Sequence ID for Program suspend command.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.98 Reference programResumeLut

Vendor specific: Reference to the LUT Sequence ID for Program resume command.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.99 Container resetSettings

Vendor specific: Container related to software reset command, for resettings the flash device.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.100 Parameter resetCmdCount

Vendor specific: Number of commands in the reset sequence

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	255
min	1

4.101 Reference resetCmdLut

Vendor specific: Reference to the LUT Sequence ID for the first command from the reset sequence.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.102 Container initResetSettings

Vendor specific: Container related to software reset command, for resettings the flash device. This reset procedure applies only at driver intialization. It might be different from the normal reset command, depending on the initial state of the flash. If not, set the same as reset command.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.103 Parameter resetCmdCount

Vendor specific: Number of commands in the reset sequence

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	255
min	1

4.104 Reference resetCmdLut

Vendor specific: Reference to the LUT Sequence ID for the first command from the reset sequence.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.105 Container initConfiguration

Vendor specific: This container describes the list of operations which must be performed at initialization time to bring the memory in the desired operating state.

Example: activate XPI mode, activate 4-byte addressing.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	255
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD

4.106 Parameter opType

Vendor specific: Operation type can be one of the following:

QSPI_IP_OP_TYPE_CMD - Simple command

QSPI_IP_OP_TYPE_WRITE_REG - Write value in external flash register

QSPI_IP_OP_TYPE_RMW_REG - RMW command on external flash register

QSPI_IP_OP_TYPE_READ_REG - Read external flash register until expected value is read

QSPI_IP_OP_TYPE_QSPI_CFG - Re-configure QSPI controller

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_OP_TYPE_CMD
literals	['QSPI_IP_OP_TYPE_CMD', 'QSPI_IP_OP_TYPE_WRITE_REG', 'QS↩ PI_IP_OP_TYPE_RMW_REG', 'QSPI_IP_OP_TYPE_READ_REG', 'Q↩ SPI_IP_OP_TYPE_QSPI_CFG']

4.107 Parameter addr

Vendor specific: Address, if used in command

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.108 Parameter size

Vendor specific: Size in bytes of configuration register, where it applies.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	4
min	1

4.109 Parameter shift

Vendor specific: Offset of configuration field, where it applies.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	32
min	0

4.110 Parameter width

Vendor specific: Witdh of configuration field, where it applies.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	32
min	0

4.111 Parameter value

Vendor specific: Value to set/expect in the bit-field.

Property	Value
type	ECUC-INTEGER-PARAM-DEF

Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	4294967295
min	0

4.112 Reference command1Lut

Vendor specific: Index of first command sequence in Lut; for RMW type this is the read command

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.113 Reference command2Lut

Vendor specific: Index of second command sequence in Lut, only used for RMW type, this is the write command.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.114 Reference weLut

Vendor specific: Index of write enable command, if needed before a write command. Only used for Write and RMW operations.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg/FlsLUT

4.115 Reference ctrlCfgPtr

Vendor specific: Reference to configuration which will be used for initializing the controller.

Valid only for QSPI_IP_OP_TYPE_QSPI_CFG operations

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/ControllerCfg

4.116 Container FlsLUT

Vendor specific: Container for the configuration of the Look Up Table holding all the Instruction/Operands sequences.

A sequence consists of a series of up to 8 instruction/operands pairs, which can ocupy up to 4 LUTs,

which are executed whenever a command is triggered to the external flash memory.

Included subcontainers:

- FlsInstructionOperandPair

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	65534
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD

4.117 Parameter FlsLUTIndex

Vendor specific: Fls LUT Index is an invariant index, used to order the LUT entries and loop

over them in the correct, configured order. Its value should be equal with the position of the

configured LUT inside the configured LUT list (the same value as the shown index).

Rationale: The generated .epc configuration might reorder the LUT elements (alphabetically), thus the default index parameter

changes, becoming out of sync with the real intended order (the values are not generated in the intended order, they are sorted).

Range:

min = 0

max = 65534

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	65534
min	0

4.118 Container FlsInstructionOperandPair

Vendor specific: One command set which holds one memory command-operand pair.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD

4.119 Parameter FlsInstrOperPairIndex

Vendor specific: Fls Instruction Operand Pair Index is an invariant index, used to order the Instr.Oper. entries and loop over them in the correct, configured order. Its value should be equal with the position of the

configured pair inside the configured pair list (the same value as the shown index).

Rationale: The generated .epc configuration might reorder the instr.oper. pairs (alphabetically), thus the index parameter

changes, becoming out of sync with the real intended order (the values are not generated in the intended order, they are sorted).

Range:

min = 0

max = 65534

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	65534
min	0

4.120 Parameter FlsLUTInstruction

Vendor specific: The instruction type used to identify the command used by the QSPI IP when

sending the command to the memory.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

Property	Value
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_LUT_INSTR_CMD
literals	['QSPI_IP_LUT_INSTR_CMD', 'QSPI_IP_LUT_INSTR_ADDR', 'QSPI_↩ IP_LUT_INSTR_DUMMY', 'QSPI_IP_LUT_INSTR_MODE', 'QSPI_IP_↩ LUT_INSTR_MODE2', 'QSPI_IP_LUT_INSTR_MODE4', 'QSPI_IP_LU↩ T_INSTR_READ', 'QSPI_IP_LUT_INSTR_WRITE', 'QSPI_IP_LUT_IN↩ STR_JMP_ON_CS', 'QSPI_IP_LUT_INSTR_ADDR_DDR', 'QSPI_IP_L↩ UT_INSTR_MODE_DDR', 'QSPI_IP_LUT_INSTR_MODE2_DDR', 'QS↩ PI_IP_LUT_INSTR_MODE4_DDR', 'QSPI_IP_LUT_INSTR_READ_D↩ DR', 'QSPI_IP_LUT_INSTR_WRITE_DDR', 'QSPI_IP_LUT_INSTR_C↩ MD_DDR', 'QSPI_IP_LUT_INSTR_CADDR', 'QSPI_IP_LUT_INSTR_↩ CADDR_DDR', 'QSPI_IP_LUT_INSTR_STOP']

4.121 Parameter FlsLUTPad

Vendor specific: Number of pads/pins used for the current command.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_LUT_PADS_1
literals	['QSPI_IP_LUT_PADS_1', 'QSPI_IP_LUT_PADS_2', 'QSPI_IP_LUT_↩ PADS_4', 'QSPI_IP_LUT_PADS_8']

4.122 Parameter FlsLUTOperand

Vendor specific: The operand of the instruction command sent to memory.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP

Property	Value
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	255
min	0

4.123 Container FlsController

Vendor specific: Container for the configuration of the avaialable QSPI controllers.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE

4.124 Parameter ControllerName

Vendor specific: The name of the configured harwdare unit name. The configured parameters will apply to this hardware unit name only.

The name of the hardware unit name represents the physical hardware unit available on chip.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	FLS_QSPI_0
literals	['FLS_QSPI_0']

4.125 Reference FlsControllerCfgRef

Vendor specific: Reference to configuration which will be used for initializing the controller.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/ControllerCfg

4.126 Container FlsMem

Vendor specific: Container for the configuration of the avaialable external flash memory hardware units.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE

4.127 Parameter FlsMemName

Vendor specific: The name of the configured flash device. The configured parameters will apply to this device only.

Property	Value
type	ECUC-STRING-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	Device_0

4.128 Parameter MemAlignment

Vendor specific: The address alignment required by the external flash (1, 2 or 4 bytes, ...), needed in the OCTA DTR Mode (DOPI).

For read operation:

- The driver will decrease the address if it is not aligned, and increasing the size to compensate.

- After the actual read, the driver ignores the first few bytes before starting the copy/comparison to the user data.

For write operation: send extra data with FFh to overwrite the overlapping memory area

? If there is a need to program from odd starting address, keep the even input address and the input data shall start with FFh.

? If there is a need to program with odd ending address, simply provide extra data with FFh in the last falling edge of clock.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

Property	Value
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	16
min	1

4.129 Parameter AHBReadEnable

Vendor specific: When set, Qspi_Ip_AhbReadEnable() will be called from Fls_Init() to allow reads via AHB.

The application can read directly through Flash memory devices address mapping.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.130 Parameter FlsMemUseSfdp

Vendor specific: Select this option to attempt auto-configuration using the information read from the SFDP table

This only works for flash devices which support the SFDP feature.

SFDP (Serial Flash Discoverable Parameters) is a JEDEC standard - JESD216D.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.131 Parameter connectionType

Vendor specific: Connection type of the flash device to the controller:

QSPI_IP_SIDE_A1	- Serial flash connected on side A1
QSPI_IP_SIDE_A2	- Serial flash connected on side A2
QSPI_IP_SIDE_B1	- Serial flash connected on side B1
QSPI_IP_SIDE_B2	- Serial flash connected on side B2

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_SIDE_A1
literals	['QSPI_IP_SIDE_A1', 'QSPI_IP_SIDE_A2', 'QSPI_IP_SIDE_B1', 'QSPI↩ _IP_SIDE_B2']

4.132 Reference FlsMemCfgRef

Vendor specific: Reference to configuration which will be used for initializing the flash memory device.

Property	Value
type	ECUC-REFERENCE-DEF

Property	Value
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/MemCfg

4.133 Reference qspiInstance

Vendor specific: QSPI controller instance to which this flash device is connected.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/FlsController

4.134 Container FlsSectorList

List of flashable sectors and pages.

Included subcontainers:

- FlsSector

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.135 Container FlsSector

Configuration description of a flashable sector

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD

4.136 Parameter FlsSectorIndex

Vendor specific: Fls Sector Index is an invariant index, used to order flash sectors and loop

over them in the correct, configured order. Its value should be equal with the position of the

configured sector inside the configured sector list (the same value as the shown index).

Rationale: The generated .epc configuration might reorder the flash sectors(alphabetically), thus the index parameter

changes, becoming out of sync with the real intended order (for example: Fls Sector Start Addresses).

Range:

min = 0

max = 65534

Property	Value
type	ECUC-INTEGER-PARAM-DEF

Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	65534
min	0

4.137 Parameter FlsPhysicalSector

Vendor specific: Physical flash device sector.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	FLS_DATA_ARRAY_0_BLOCK_3_S000

Property	Value
literals	['FLS_DATA_ARRAY_0_BLOCK_3_S000', 'FLS_DATA_ARRAY_0_BLOCK_3_S001', 'FLS_DATA_ARRAY_0_BLOCK_3_S002', 'FLS_DATA_ARRAY_0_BLOCK_3_S003', 'FLS_DATA_ARRAY_0_BLOCK_3_S004', 'FLS_DATA_ARRAY_0_BLOCK_3_S005', 'FLS_DATA_ARRAY_0_BLOCK_3_S006', 'FLS_DATA_ARRAY_0_BLOCK_3_S007', 'FLS_DATA_ARRAY_0_BLOCK_3_S008', 'FLS_DATA_ARRAY_0_BLOCK_3_S009', 'FLS_DATA_ARRAY_0_BLOCK_3_S010', 'FLS_DATA_ARRAY_0_BLOCK_3_S011', 'FLS_DATA_ARRAY_0_BLOCK_3_S012', 'FLS_DATA_ARRAY_0_BLOCK_3_S013', 'FLS_DATA_ARRAY_0_BLOCK_3_S014', 'FLS_DATA_ARRAY_0_BLOCK_3_S015', 'FLS_DATA_ARRAY_0_BLOCK_3_S016', 'FLS_DATA_ARRAY_0_BLOCK_3_S017', 'FLS_DATA_ARRAY_0_BLOCK_3_S018', 'FLS_DATA_ARRAY_0_BLOCK_3_S019', 'FLS_DATA_ARRAY_0_BLOCK_3_S020', 'FLS_DATA_ARRAY_0_BLOCK_3_S021', 'FLS_DATA_ARRAY_0_BLOCK_3_S022', 'FLS_DATA_ARRAY_0_BLOCK_3_S023', 'FLS_DATA_ARRAY_0_BLOCK_3_S024', 'FLS_DATA_ARRAY_0_BLOCK_3_S025', 'FLS_DATA_ARRAY_0_BLOCK_3_S026', 'FLS_DATA_ARRAY_0_BLOCK_3_S027', 'FLS_DATA_ARRAY_0_BLOCK_3_S028', 'FLS_DATA_ARRAY_0_BLOCK_3_S029', 'FLS_DATA_ARRAY_0_BLOCK_3_S030', 'FLS_DATA_ARRAY_0_BLOCK_3_S031', 'FLS_DATA_ARRAY_0_BLOCK_3_S032', 'FLS_DATA_ARRAY_0_BLOCK_3_S033', 'FLS_DATA_ARRAY_0_BLOCK_3_S034', 'FLS_DATA_ARRAY_0_BLOCK_3_S035', 'FLS_DATA_ARRAY_0_BLOCK_3_S036', 'FLS_DATA_ARRAY_0_BLOCK_3_S037', 'FLS_DATA_ARRAY_0_BLOCK_3_S038', 'FLS_DATA_ARRAY_0_BLOCK_3_S039', 'FLS_DATA_ARRAY_0_BLOCK_3_S040', 'FLS_DATA_ARRAY_0_BLOCK_3_S041', 'FLS_DATA_ARRAY_0_BLOCK_3_S042', 'FLS_DATA_ARRAY_0_BLOCK_3_S043', 'FLS_DATA_ARRAY_0_BLOCK_3_S044', 'FLS_DATA_ARRAY_0_BLOCK_3_S045', 'FLS_DATA_ARRAY_0_BLOCK_3_S046', 'FLS_DATA_ARRAY_0_BLOCK_3_S047', 'FLS_DATA_ARRAY_0_BLOCK_3_S048', 'FLS_DATA_ARRAY_0_BLOCK_3_S049', 'FLS_DATA_ARRAY_0_BLOCK_3_S050', 'FLS_DATA_ARRAY_0_BLOCK_3_S051', 'FLS_DATA_ARRAY_0_BLOCK_3_S052', 'FLS_DATA_ARRAY_0_BLOCK_3_S053', 'FLS_DATA_ARRAY_0_BLOCK_3_S054', 'FLS_DATA_ARRAY_0_BLOCK_3_S055', 'FLS_DATA_ARRAY_0_BLOCK_3_S056', 'FLS_DATA_ARRAY_0_BLOCK_3_S057', 'FLS_DATA_ARRAY_0_BLOCK_3_S058', 'FLS_DATA_ARRAY_0_BLOCK_3_S059', 'FLS_DATA_ARRAY_0_BLOCK_3_S060', 'FLS_DATA_ARRAY_0_BLOCK_3_S061', 'FLS_DATA_ARRAY_0_BLOCK_3_S062', 'FLS_DATA_ARRAY_0_BLOCK_3_S063', 'FLS_DATA_ARRAY_0_BLOCK_3_S064', 'FLS_DATA_ARRAY_0_BLOCK_3_S065', 'FLS_DATA_ARRAY_0_BLOCK_3_S066', 'FLS_DATA_ARRAY_0_BLOCK_3_S067', 'FLS_DATA_ARRAY_0_BLOCK_3_S068', 'FLS_DATA_ARRAY_0_BLOCK_3_S069', 'FLS_DATA_ARRAY_0_BLOCK_3_S070', 'FLS_DATA_ARRAY_0_BLOCK_3_S071', 'FLS_DATA_ARRAY_0_BLOCK_3_S072', 'FLS_DATA_ARRAY_0_BLOCK_3_S073', 'FLS_DATA_ARRAY_0_BLOCK_3_S074', 'FLS_DATA_ARRAY_0_BLOCK_3_S075', 'FLS_DATA_ARRAY_0_BLOCK_3_S076', 'FLS_DATA_ARRAY_0_BLOCK_3_S077', 'FLS_DATA_ARRAY_0_BLOCK_3_S078', 'FLS_DATA_ARRAY_0_BLOCK_3_S079', 'FLS_DATA_ARRAY_0_BLOCK_3_S080', 'FLS_DATA_ARRAY_0_BLOCK_3_S081', 'FLS_DATA_ARRAY_0_BLOCK_3_S082', 'FLS_DATA_ARRAY_0_BLOCK_3_S083', 'FLS_DATA_ARRAY_0_BLOCK_3_S084', 'FLS_DATA_ARRAY_0_BLOCK_3_S085', 'FLS_DATA_ARRAY_0_BLOCK_3_S086', 'FLS_DATA_ARRAY_0_BLOCK_3_S087', 'FLS_DATA_ARRAY_0_BLOCK_3_S088', 'FLS_DATA_ARRAY_0_BLOCK_3_S089', 'FLS_DATA_ARRAY_0_BLOCK_3_S090', 'FLS_DATA_ARRAY_0_BLOCK_3_S091', 'FLS_DATA_ARRAY_0_BLOCK_3_S092', 'FLS_DATA_ARRAY_0_BLOCK_3_S093', 'FLS_DATA_ARRAY_0_BLOCK_3_S094', 'FLS_DATA_ARRAY_0_BLOCK_3_S095', 'FLS_DATA_

Property	Value

4.138 Parameter FlsNumberOfSectors

Number of continuous sectors with the above characteristics.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	65535
min	1

4.139 Parameter FlsPageSize

Size of one page of this sector. Implementation Type: Fls_LengthType.

For internal flash, page size is 8 byte

For external flash, page size is chip specific.

For example: In Macronix devices, the ECC algorithm uses a Hamming code that can correct a single bit error per 16-Byte page.

It is recommended that data be programmed in multiples of 16 bytes using the Page Program command instead of programming a byte or a word at a time using the Program command.

Each group of 16 bytes must fall within the same 16-Byte boundary.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	8
max	4294967295
min	0

4.140 Parameter FlsSectorSize

Size of this sector. Implementation Type: Fls_LengthType.

Note: Size of the sector should be a multiple of the page size.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	4096
max	4294967295
min	0

4.141 Parameter FlsSectorStartaddress

Start address of this sector.

Implementation Type: Fls_AddressType.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC

Property	Value
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.142 Parameter FlsSectorEraseAsynch

Vendor specific: Enable asynchronous execution of the erase job in the Fls_MainFunction function which doesn't wait (block)

for completion of the sector erase operation. The flash driver doesn't use the erase access code to the erase flash sector

in asynchronous mode so it can be used only on flash sectors which belong to flash array different from flash array the

application is executing from.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.143 Parameter FlsPageWriteAsynch

Vendor specific: Enable asynchronous execution of the write job in the Fls_MainFunction function which doesn't wait (block) for completion of the page write operation(s). The flash driver doesn't use the write access code to the write flash page(s)

in asynchronous mode so it can be used only on flash sectors which belong to flash array different from flash array the

application is executing from.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.144 Parameter FlsHwCh

Vendor specific: The hardware channel type of the current sector: internal flash or external QSPI flash sector.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	FLS_CH_INTERN
literals	['FLS_CH_INTERN', 'FLS_CH_QSPI']

4.145 Parameter FlsSectorHwAddress

Vendor specific: Hardware address of this sector, as needed by the external flash device (usually starting from 0).

Applicable only to external sectors. This value is used to access the hardware sector on the attached device

and will be sent as parameter of flash comands, so it should be completed to meet the requirements of the

external flash memory type and configured operating mode.

Internally, this address is added to the MCU base addresses of each channel, configured in SF{A/B}{1/2}AD
registers, in order

to select the corresponding external device hw channel.

Example: FlsSectorHwAddress = 0x100

Sector hardware channel = FLS_CH_EXTERN_QSPI_0_A2

FlsSerialFlashA1TopAddr = 0x24000000

The address used by the driver internally will be 0x24000100, thus selecting external

flash device A2 and accessing internal location 0x100 of the memory.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.146 Reference flashInstance

Vendor specific: External flash device instance to which this sector belongs.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false

Property	Value
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Fls/FlsConfigSet/FlsExternalDriver/FlsMem

4.147 Container AutosarExt

Vendor specific: This container contains the global Non-Autosar configuration parameters of the Fls driver.

This container is a MultipleConfigurationContainer, i.e. this container and

its sub-containers exist once per configuration set.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.148 Parameter FlsEnableUserModeSupport

Vendor specific: When this parameter is enabled, the FLS module will adapt to run from User Mode, with the following measures:

configuring REG_PROT for Fls IPs so that the registers under protection can be accessed from user mode by setting UAA bit in REG_PROT_GCR to 1

for more information and availability on this platform, please see chapter User Mode Support in IM

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP

Property	Value
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.149 Parameter FlsQspiLockLUT

Vendor specific: Enable the Lock/Unlock of the LUT for the external QuadSPI memory.

If Enabled, the LUT is unlocked at the beginning of the Init phase and locked at the end of it.

If Disabled, the LUT has to be unlocked if the Init phase is supposed to populate it.

Note: not used in the driver code.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.150 Parameter FlsSynchronizeCache

Vendor specific:

Synchronize the memory by invalidating the cache after each flash hardware operation.

The FLS driver needs to maintain the memory coherency by means of three methods:

1. Disable data cache, or
2. Configure the flash region upon which the driver operates, as non-cacheable, or
3. Enable the FlsSynchronizeCache feature.

Depending on the application configuration, one option may be more beneficial than other.

Enabled: The FLS driver will call Mcl cache API functions in order to invalidate the cache

after each high voltage operation(write,erase) and before each read operation, in order

to ensure that the cache and the modified flash memory are in sync.

If enabled, the driver will attempt to invalidate only the modified lines from the cache.

If the size of the region to be invalidated is greater than half of the cache size, then

the entire cache is invalidated.

Note: If enabled, the MclLmemEnableCacheApi parameter has to be enabled and the MCL plugin included as a dependency.

Disabled: The upper layers have to ensure that the flash region upon which the driver operates is not cached.

This can be obtained by either disabling the data cache or by configuring the memory region as non-cacheable.

Note: This feature is applicable only if supported on the current platform.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.151 Parameter FlsInvalidPrefetchBufFromRam

Vendor specific: Allow to allocate the function invalidation the prefetch buffer of internal flash on RAM or FLASH.

Rationale: On some platforms the RAM may not be executable due to security restrictions.

If Enabled: the invalidate prefetch buffer functionality is executable from RAM.

If Disabled: the invalidate prefetch buffer functionality is executable from FLASH.

Note: For more information please see the chapter Tips for FLS integration in the Integration Manual.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.152 Parameter FlsInternalSectorsConfigured

Vendor specific:

Boolean parameter which must be enabled if internal flash sectors are configured.

Enabled: At least one internal flash sector is configured in any variant.

Disabled: No internal flash sector is configured in any variant, only external flash sectors are present.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.153 Parameter FlsExternalSectorsConfigured

Vendor specific:

Boolean parameter which must be enabled if external flash sectors are configured.

Enabled: At least one external flash sector is configured in any variant.

Disabled: No external flash sector is configured in any variant, only internal flash sectors are present.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.154 Container FlsGeneral

Container for general parameters of the flash driver. These parameters are always pre-compile.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.155 Parameter FlsEnableDevAssert

Vendor specific:

true: Development error checking at IP level is enabled.

false: Development error checking at IP level is disabled.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.156 Parameter FlsAcLoadOnJobStart

The flash driver shall load the flash access code to RAM whenever an erase or write job is started and unload (overwrite) it after that job has been finished or canceled.

true: Flash access code loaded on job start / unloaded on job end or error.

false: Flash access code not loaded to / unloaded from RAM at all.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.157 Parameter FlsCleanCacheAfterLoadAc

Vendor specific: Pre-processor switch to allow to clean cache after loading AccessCode to RAM to ensure the synchronization between cache and RAM memory.

This action might be needed in case the AccessCode function is coppied to a cacheable area.

true: cleans cache after loading AccessCode function to RAM to write cache data to the actual RAM memory

false: does not clean cache after loading AccessCode function to RAM

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.158 Parameter FlsBaseAddress

The flash memory start address (see also FLS118).

FLS169: This parameter defines the lower boundary for read / write / erase and compare jobs.Note:Not needed / supported by the driver.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.159 Parameter FlsBlankCheckApi

Compile switch to enable and disable the Fls_BlankCheck function.

true: API supported / function provided.

false: API not supported / function not provided

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.160 Parameter FlsCancelApi

Compile switch to enable and disable the Fls_Cancel function.

true: API supported / function provided.

false: API not supported / function not provided

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.161 Parameter FlsCompareApi

Compile switch to enable and disable the Fls_Compare function.

true: API supported / function provided.

false: API not supported / function not provided

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.162 Parameter FlsDevErrorDetect

Pre-processor switch to enable and disable development error detection.

true: Development error detection enabled.

false: Development error detection disabled.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.163 Parameter FlsDriverIndex

Index of the driver, used by FEE.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	true
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	254
min	0

4.164 Parameter FlsGetJobResultApi

Compile switch to enable and disable the Fls_GetJobResult function.

true: API supported / function provided.

false: API not supported / function not provided

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.165 Parameter FlsGetStatusApi

Compile switch to enable and disable the Fls_GetStatus function.

true: API supported / function provided.

false: API not supported / function not provided

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.166 Parameter FlsSetModeApi

Compile switch to enable and disable the Fls_SetMode function.

true: API supported / function provided.

false: API not supported / function not provided

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.167 Parameter FlsTotalSize

The total amount of flash memory in bytes (see also FLS118). FLS170: This parameter in conjunction with FLS_BASE_ADDRESS defines the upper boundary for read / write / erase and compare jobs.

Note:Not needed / supported by the driver.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.168 Parameter FlsUseInterrupts

Not supported by the driver.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.169 Parameter FlsVersionInfoApi

Pre-processor switch to enable / disable the API to read out the modules version information.

true: Version info API enabled.

false: Version info API disabled.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.170 Parameter FlsECCCheck

Vendor specific: Pre-processor switch to enable / disable the API to report data storage (ECC) errors to the flash driver.

This is the first ECC handling approach which modifies the program counter to skip the instruction causing the fault.

Please read the chapter Exception Handler in case of ECC error in IM for more information.

true : The ECC check by HardfaultHandler API is enabled.

false: The ECC check by HardfaultHandler API is disabled.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.171 Parameter FlsECCHandlingProtectionHook

Vendor specific: Pre-processor switch to enable / disable the API to report data storage (ECC) errors to the flash driver.

This is the second ECC handling approach which is compatible with Autosar Os.

Please read the chapter Exception Handler in case of ECC error in IM for more information.

true : The ECC check by AutosarOs API is enabled.

false: The ECC check by AutosarOs API is disabled.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.172 Parameter FlsEraseVerificationEnabled

Pre-processor switch to enable / disable the erase blank check. After a flash block has been erased, the erase blank check compares the contents of the addressed memory area against the value of an erased flash cell to check that the block has been completely erased.

true: Memory region is checked to be erased.

false: Memory region is not checked to be erased.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

Property	Value
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.173 Parameter FlsWriteVerificationEnabled

Pre-processor switch to enable / disable the write verify check. After writing a flash block, the write verify check compares the contents of the reprogrammed memory area against the contents of the provided application buffer to check that the block has been completely reprogrammed.

true: Written data is compared directly after write.

false: Written date is not compared directly after write.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.174 Parameter FlsMaxEraseBlankCheck

Vendor specific: The maximum number of bytes to blank check in one cycle of the flash driver's job processing function. Affects only the flash blocks that have enabled asynchronous execution of the erase job (FlsSectorEraseAsynch=true)

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	256
max	65536
min	8

4.175 Parameter FlsTimeoutSupervisionEnabled

Compile switch to enable timeout supervision.

true: timeout supervision for read/erase/write/compare jobs enabled.

false: timeout supervision for read/erase/write/compare jobs disabled.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.176 Parameter FlsTimeoutMethod

Vendor specific: Counter type used in timeout detection for FLS service request.

Based on selected counter type the timeout value will be interpreted as follows:

OSIF_COUNTER_DUMMY - Counts the number of iterations of the waiting loop. The actual timeout depends on many factors: operation type, compiler optimizations, interrupts or other tasks in the system, etc.

OSIF_COUNTER_SYSTEM - Microseconds.

OSIF_COUNTER_CUSTOM - Defined by user implementation of timing services

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	OSIF_COUNTER_DUMMY
literals	['OSIF_COUNTER_DUMMY', 'OSIF_COUNTER_SYSTEM', 'OSIF_COU↩ NTER_CUSTOM']

4.177 Parameter FlsAsyncWriteTimeout

Vendor specific: Fls Async Write Timeout is the timeout value for write operation in asynchronous mode.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	2147483647
max	2147483647
min	0

4.178 Parameter FlsAsyncEraseTimeout

Vendor specific: Fls Async Erase Timeout is the timeout value for erase operation in asynchronous mode.

Property	Value
type	ECUC-INTEGER-PARAM-DEF

Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	2147483647
max	2147483647
min	0

4.179 Parameter FlsSyncWriteTimeout

Vendor specific: Fls Sync Write Timeout is the timeout value for write operation in synchronous mode.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	2147483647
max	2147483647
min	0

4.180 Parameter FlsSyncEraseTimeout

Vendor specific: Fls Sync Erase Timeout is the timeout value for erase operation in synchronous mode.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	2147483647
max	2147483647
min	0

4.181 Parameter FlsAbortTimeout

Vendor specific: Fls Abort Timeout is the timeout value for aborting an ongoing operation.

The timeout is used also in Fls_Cancel API and Abort Erase suspend, if enabled and if the flash hardware channel does not support an immediate abort feature.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	32767
max	2147483647
min	0

4.182 Parameter FlsQspiIpTimeoutOsifCounterType

Vendor specific: Counter type used in timeout detection for QSPI operations.

Based on selected counter type the timeout value will be interpreted as follows:

OSIF_COUNTER_DUMMY - Counts the number of iterations of the waiting loop. The actual timeout depends on many factors: operation type, compiler optimizations, interrupts or other tasks in the system, etc.

OSIF_COUNTER_SYSTEM - Microseconds.

OSIF_COUNTER_CUSTOM - Defined by user implementation of timing services

Note: Qspi always uses timeout

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	OSIF_COUNTER_DUMMY
literals	['OSIF_COUNTER_DUMMY', 'OSIF_COUNTER_SYSTEM', 'OSIF_COU↩ NTER_CUSTOM']

4.183 Parameter FlsQspiSyncReadTimeout

Vendor specific: Fls Qspi Sync Read Timeout is the timeout value for read operation.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	2147483647
max	2147483647
min	0

4.184 Parameter FlsQspiAsyncWriteTimeout

Vendor specific: Fls Qspi Async Write Timeout is the timeout value for QSPI write operation in asynchronous mode.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	2147483647
max	2147483647
min	0

4.185 Parameter FlsQspiAsyncEraseTimeout

Vendor specific: Fls Qspi Async Erase Timeout is the timeout value for QSPI erase operation in asynchronous mode.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	2147483647
max	2147483647
min	0

4.186 Parameter FlsQspiSyncWriteTimeout

Vendor specific: Fls Qspi Sync Write Timeout is the timeout value for QSPI write operation in synchronous mode.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	2147483647
max	2147483647
min	0

4.187 Parameter FlsQspiSyncEraseTimeout

Vendor specific: Fls Qspi Sync Erase Timeout is the timeout value for QSPI erase operation in synchronous mode.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	2147483647
max	2147483647
min	0

4.188 Parameter FlsQspiCommandCompleteTimeout

Vendor specific: Fls Qspi Command Complete Timeout is the timeout value for waiting for a QSPI command to be completed.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	2147483647
min	0

4.189 Parameter FlsQspiResetTimeout

Vendor specific: Fls Qspi Reset Timeout is the timeout for waiting for the external device to become available after a software reset.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	2147483647
min	0

4.190 Parameter FlsQspiFlashInitTimeout

Vendor specific: Fls Flash Init Timeout is the timeout for completing the initializtion operation sequence for the external flash at startup.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	2147483647
min	0

4.191 Parameter FlsQspiSoftwareResetDelay

Vendor specific: Fls Qspi Reset Delay is the waiting time after changing the value of the QSPI software reset bits in MCR register.

Note: The default value is calculated in the number of CPU cycles for the worst case scenario (with maximum possible CPU frequency and minimum possible flash clock frequency).

See the note of MCR register of QSPI chapter in Reference manual for more information.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	2147483647
min	0

4.192 Parameter FlsQspiTxBufferResetDelay

Vendor specific: Fls Qspi TX Buffer Reset Delay is the waiting time after changing the value of the QSPI TX FIFO/buffer reset bits in MCR register.

Note: The default value is calculated in the number of CPU cycles for the worst case scenario (with maximum possible CPU frequency and minimum possible flash clock frequency).

See the note of MCR register of QSPI chapter in Reference manual for more information.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	2147483647
min	0

4.193 Parameter FlsQspiWriteEnableRetries

Vendor specific: Number of attempts when sending the Write Enable command to the external flash.

The driver will read back the status register after each attempt and check the Busy bit.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	3
max	100
min	0

4.194 Reference FlsEcucPartitionRef

Maps the Flash driver to zero or one ECUC partition to make the driver API available in this partition.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/EcuC/EcucPartitionCollection/EcucPartition

4.195 Container FlsPublishedInformation

Additional published parameters not covered by CommonPublishedInformation container.

Note that these parameters do not have any configuration class setting, since they are published information.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.196 Parameter FlsAcLocationErase

Position in RAM, to which the erase flash access code has to be loaded.

Only relevant if the erase flash access code is not position independent. If this information is not provided it is assumed that the erase flash access code is position independent and that therefore the RAM position can be freely configured.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	4294967295
min	0

4.197 Parameter FlsAcLocationWrite

Vendor specific: Position in RAM, to which the write flash access code has to be loaded.

Only relevant if the write flash access code is not position independent. If this information is not provided it is assumed that the write flash access code is position independent and that therefore the RAM position can be freely configured.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	4294967295
min	0

4.198 Parameter FlsAcSizeErase

Number of bytes in RAM needed for the erase flash access code.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	4294967295
min	0

4.199 Parameter FlsAcSizeWrite

Number of bytes in RAM needed for the write flash access code.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	4294967295
min	0

4.200 Parameter FlsEraseTime

Maximum time to erase one complete flash sector [sec].

Note:This value can be found on DS as the maximum erase time occurs after the specified number of program/erase cycles .

Property	Value
type	ECUC-FLOAT-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	5.0
max	5.0
min	0.0

4.201 Parameter FlsErasedValue

The contents of an erased flash memory cell.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	4294967295
max	4294967295
min	0

4.202 Parameter FlsECCValue

Vendor specific: The contents of an ECC flash memory line.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP

Property	Value
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	1427461397
max	4294967295
min	0

4.203 Parameter FlsExpectedHwId

Unique identifier of the hardware device that is expected by this driver (the device for which this driver has been implemented).

Only relevant for external flash drivers.

Property	Value
type	ECUC-STRING-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0

4.204 Parameter FlsSpecifiedEraseCycles

Number of erase cycles specified for the flash device (usually given in the device data sheet).

FLS198: If the number of specified erase cycles depends on the operating environment (temperature, voltage, ...) during reprogramming of the flash device, the minimum number for which a data retention of at least 15 years over the temperature range from -40C .. +125C can be guaranteed shall be given.

Note:If there are different numbers of specified erase cycles for different flash sectors of the device this parameter has to be extended to a parameter list (similar to the sector list above).

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	100000
max	4294967295
min	0

4.205 Parameter FlsWriteTime

Maximum time to program one complete flash page [sec].

Property	Value
type	ECUC-FLOAT-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	5.0E-4
max	5.0E-4
min	0.0

4.206 Container CommonPublishedInformation

Common container, aggregated by all modules. It contains published information about vendor and versions.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.207 Parameter ArReleaseMajorVersion

Vendor specific: Major version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	4
max	4
min	4

4.208 Parameter ArReleaseMinorVersion

Vendor specific: Minor version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

Property	Value
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	4
max	4
min	4

4.209 Parameter ArReleaseRevisionVersion

Vendor specific: Patch version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.210 Parameter ModuleId

Vendor specific: Module ID of this module.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false

Property	Value
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	92
max	92
min	92

4.211 Parameter SwMajorVersion

Vendor specific: Major version number of the vendor specific implementation of the module. The numbering is vendor specific.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	1
max	1
min	1

4.212 Parameter SwMinorVersion

Vendor specific: Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false

Property	Value
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.213 Parameter SwPatchVersion

Vendor specific: Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	1
max	1
min	1

4.214 Parameter VendorApiInfix

Vendor specific: In driver modules which can be instantiated several times on a single ECU, BSW00347 requires that the name of APIs is extended by the VendorId and a vendor specific name.

This parameter is used to specify the vendor specific name. In total, the implementation specific name is generated as follows:

<ModuleName>_<VendorId>_<VendorApiInfix>.

E.g. assuming that the VendorId of the implementor is 123 and the implementer chose a VendorApiInfix of "v11r456" a api name Can_Write defined in the SWS will translate to Can_123_v11r456Write.

This parameter is mandatory for all modules with upper multiplicity > 1. It shall not be used for modules with upper multiplicity =1.

Property	Value
type	ECUC-STRING-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	

4.215 Parameter VendorId

Vendor specific: Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	43
max	43
min	43

Chapter 5

Module Index

5.1 Software Specification

Here is a list of all modules:

Chapter 6

Module Documentation

6.1 FLS Driver

6.1.1 Detailed Description

Data Structures

- struct Fls_Flash_InternalSectorInfoType

  FLASH physical sector description. More...
- struct Fls_QspiCfgConfigType

  Fls Qspi CfgConfig Type. More...
- struct Fls_ConfigType

  Fls Config Type. More...

Macros

- #define FLS_DEVICE_INSTANCE_INVALID
- #define FLS_API_VENDOR_ID

  Version Check parameters.
- #define FLS_MODULE_ID

  AUTOSAR module identification.
- #define FLS_INSTANCE_ID

  AUTOSAR module instance identification.
- #define FLS_E_PARAM_CONFIG

  Development error codes (passed to DET).
- #define FLS_E_PARAM_ADDRESS

  API service called with wrong address parameter.
- #define FLS_E_PARAM_LENGTH

  API service called with wrong length parameter.
- #define FLS_E_PARAM_DATA

  API service called with wrong data parameter.

- #define FLS_E_UNINIT

  API service called without module initialization.
- #define FLS_E_BUSY

  API service called while driver still busy.
- #define FLS_E_PARAM_POINTER

  API service called with NULL pointer.
- #define FLS_E_VERIFY_ERASE_FAILED

  Runtime error codes (passed to DET).
- #define FLS_E_VERIFY_WRITE_FAILED

  Write verification (compare) failed.
- #define FLS_E_TIMEOUT

  Timeout exceeded.
- #define FLS_E_ERASE_FAILED

  Transient Faults codes (passed to DET).
- #define FLS_E_WRITE_FAILED

  Flash write failed (HW)
- #define FLS_E_READ_FAILED

  Flash read failed (HW)
- #define FLS_E_COMPARE_FAILED

  Flash compare failed (HW)
- #define FLS_INIT_ID

  All service IDs (passed to DET).
- #define FLS_ERASE_ID

  service ID of function: Fls_Erase. (passed to DET)
- #define FLS_WRITE_ID

  service ID of function: Fls_Write. (passed to DET)
- #define FLS_CANCEL_ID

  service ID of function: Fls_Cancel. (passed to DET)
- #define FLS_GETJOBRESULT_ID

  service ID of function: Fls_GetJobResult. (passed to DET)
- #define FLS_MAINFUNCTION_ID

  service ID of function: Fls_MainFunction. (passed to DET)
- #define FLS_READ_ID

  service ID of function: Fls_Read. (passed to DET)
- #define FLS_COMPARE_ID

  service ID of function: Fls_Compare. (passed to DET)
- #define FLS_SETMODE_ID

  service ID of function: Fls_SetMode. (passed to DET)
- #define FLS_GETVERSIONINFO_ID

  service ID of function: Fls_GetVersionInfo. (passed to DET)
- #define FLS_BLANK_CHECK_ID

  service ID of function: Fls_BlankCheck. (passed to DET)
- #define FLS_SECTOR_ERASE_ASYNCH

  All sector flags.
- #define FLS_PAGE_WRITE_ASYNCH

  fls page write asynch
- #define FLS_START_SEC_CODE

Start of Fls section CODE.
- #define FLS_STOP_SEC_CODE

  Stop of Fls section CODE.
- #define FLS_IPW_CFG_INVALID

Types Reference

- typedef uint32 Fls_SectorIndexType

  Logical sector index.
- typedef uint16 Fls_CrcType

  Fls CRC Type.
- typedef uint32 Fls_AddressType

  Fls Address Type.
- typedef uint32 Fls_LengthType

  Fls Length Type.
- typedef uint32 Fls_SectorCountType

  Fls Sector Count Type.
- typedef uint8 Fls_BlockNumberOfSectorType

  Fls BLock Count Type.
- typedef Ftfc_ConfigType Fls_InternalConfigType

  Fls Internal Flash Type.
- typedef void(∗ Fls_JobEndNotificationPtrType) (void)

  Fls Job End Notification Pointer Type.
- typedef void(∗ Fls_JobErrorNotificationPtrType) (void)

  Fls Job Error Notification Pointer Type.
- typedef void(∗ Fls_ACCallbackPtrType) (void)

  Pointer type of Fls_AC_Callback function.
- typedef void(∗ Fls_AcErasePtrType) (void(∗CallBack) (void))

  Define pointer type of erase access code function.
- typedef void(∗ Fls_AcWritePtrType) (void(∗CallBack) (void))

  Define pointer type of write access code function.
- typedef void(∗ Fls_ReadFunctionPtrType) (void)

  Pointer type of Fls_ReadFunctionPtrType function.

Enum Reference

- enum Fls_HwChType

  Flash sector channel type.
- enum Fls_JobType

  Type of job currently executed by Fls_MainFunction.
- enum Fls_LLDReturnType

  Result of low-level flash operation.
- enum Fls_LLDJobType

  Type of job currently executed by Fls_LLDMainFunction.
- enum Fls_CrcDataSizeType

  Size of data to be processeed by CRC.

Function Reference

- void Fls_MainFunction (void)

  Performs actual flash read, write, erase and compare jobs.
- void Fls_Init (const Fls_ConfigType ∗ConfigPtr)

  The function initializes Fls module.
- Std_ReturnType Fls_Write (Fls_AddressType TargetAddress, const uint8 ∗SourceAddressPtr, Fls_LengthType Length)

  Write one or more complete flash pages to the flash device.
- Std_ReturnType Fls_Erase (Fls_AddressType TargetAddress, Fls_LengthType Length)

  Erase one or more complete flash sectors.
- Std_ReturnType Fls_Read (Fls_AddressType SourceAddress, uint8 ∗TargetAddressPtr, Fls_LengthType Length)

  Reads from flash memory.

Variables

- Fls_AddressType Fls_u32JobAddrIt

  Logical address of data block currently processed by Fls_MainFunction.
- Fls_AddressType Fls_u32JobAddrEnd

  Last logical address to be processed by a job.
- volatile Fls_SectorIndexType Fls_u32JobSectorIt

  Index of flash sector currently processed by a job.
- Fls_SectorIndexType Fls_u32JobSectorEnd

  Index of last flash sector by current job.
- volatile MemIf_JobResultType Fls_eLLDJobResult

  Result of last flash hardware job.
- Fls_LLDJobType Fls_eLLDJob

  Type of current flash hardware job - used for asynchronous operating mode.
- const Fls_ConfigType ∗ Fls_pConfigPtr

  Pointer to current flash module configuration set.

6.1.2 Data Structure Documentation

6.1.2.1 struct Fls_Flash_InternalSectorInfoType

FLASH physical sector description.

Definition at line 317 of file Fls_Types.h.

Data Fields

Type	Name	Description
uint32	pSectorStartAddressPtr	FLASH physical sector start address.
uint32	u32SectorId	Corresponding number in sector location to calc cfgCRC.

6.1.2.2 struct Fls_QspiCfgConfigType

Fls Qspi CfgConfig Type.

Fls Qspi CfgConfig Type

Definition at line 329 of file Fls_Types.h.

Data Fields

- const uint8(∗ u8SectFlashUnit)[ ]

  External flash unit assigned to each sector. Size: u32SectorCount.
- const uint8 u8FlashUnitsCount

  Number of serial flash instances.
- const Qspi_Ip_MemoryConnectionType(∗ paFlashConnectionCfg)[ ]

  Connection for each external memory device to available controllers. Size: u8FlashUnitsCount.
- const uint8(∗ u8FlashConfig)[ ]

  Configuration index used for each flash unit. Size: u8FlashUnitsCount.
- const boolean(∗ paAHBReadCfg)[ ]

  AHB direct reads configurations. Size: u8FlashUnitsCount.
- const uint8 u8FlashConfigCount

  Number of serial flash configurations.
- const Qspi_Ip_MemoryConfigType(∗ paFlashCfg)[ ]

  External memory devices configurations. Size: u8FlashConfigCount.
- const uint8 u8QspiUnitsCount

  Number of QSPI hardware instances.
- const uint8(∗ u8QspiConfig)[ ]

  Configuration for each QSPI unit. Size: u8QspiUnitsCount].
- const uint8 u8QspiConfigCount

  Number of QSPI configurations.
- const Qspi_Ip_ControllerConfigType(∗ paQspiUnitCfg)[ ]

  QSPI configurations. Size: u8QspiConfigCount.

6.1.2.2.1 Field Documentation

6.1.2.2.1.1 u8SectFlashUnit `const uint8(* u8SectFlashUnit)[]`

External flash unit assigned to each sector. Size: u32SectorCount.

Definition at line 334 of file Fls_Types.h.

6.1.2.2.1.2 u8FlashUnitsCount `const uint8 u8FlashUnitsCount`

Number of serial flash instances.

Definition at line 338 of file Fls_Types.h.

6.1.2.2.1.3 paFlashConnectionCfg `const Qspi_Ip_MemoryConnectionType(* paFlashConnectionCfg)[ ]`

Connection for each external memory device to available controllers. Size: u8FlashUnitsCount.

Definition at line 342 of file Fls_Types.h.

6.1.2.2.1.4 u8FlashConfig `const uint8(* u8FlashConfig)[ ]`

Configuration index used for each flash unit. Size: u8FlashUnitsCount.

Definition at line 346 of file Fls_Types.h.

6.1.2.2.1.5 paAHBReadCfg `const boolean(* paAHBReadCfg)[ ]`

AHB direct reads configurations. Size: u8FlashUnitsCount.

Definition at line 350 of file Fls_Types.h.

6.1.2.2.1.6 u8FlashConfigCount `const uint8 u8FlashConfigCount`

Number of serial flash configurations.

Definition at line 355 of file Fls_Types.h.

6.1.2.2.1.7 paFlashCfg `const Qspi_Ip_MemoryConfigType(* paFlashCfg)[ ]`

External memory devices configurations. Size: u8FlashConfigCount.

Definition at line 359 of file Fls_Types.h.

6.1.2.2.1.8 u8QspiUnitsCount `const uint8 u8QspiUnitsCount`

Number of QSPI hardware instances.

Definition at line 364 of file Fls_Types.h.

6.1.2.2.1.9 u8QspiConfig `const uint8(* u8QspiConfig)[]`

Configuration for each QSPI unit. Size: u8QspiUnitsCount].

Definition at line 368 of file Fls_Types.h.

6.1.2.2.1.10 u8QspiConfigCount `const uint8 u8QspiConfigCount`

Number of QSPI configurations.

Definition at line 372 of file Fls_Types.h.

6.1.2.2.1.11 paQspiUnitCfg `const Qspi_Ip_ControllerConfigType(* paQspiUnitCfg)[]`

QSPI configurations. Size: u8QspiConfigCount.

Definition at line 376 of file Fls_Types.h.

6.1.2.3 struct Fls_ConfigType

Fls Config Type.

Fls module initialization data structure

Definition at line 382 of file Fls_Types.h.

Data Fields

- Fls_AcErasePtrType acErasePtr

  pointer to erase access code function in RAM or ROM
- Fls_AcWritePtrType acWritePtr

  pointer to write access code function in RAM or ROM
- Fls_ACCallbackPtrType acCallBackPtr

  pointer to ac callback function
- Fls_JobEndNotificationPtrType jobEndNotificationPtr

  pointer to job end notification function
- Fls_JobErrorNotificationPtrType jobErrorNotificationPtr

  pointer to job error notification function
- Fls_ReadFunctionPtrType FlsReadFunctionCallout

  pointer to read to flash memory callout
- MemIf_ModeType eDefaultMode

  default FLS device mode after initialization (MEMIF_MODE_FAST, MEMIF_MODE_SLOW)
- Fls_LengthType u32MaxReadFastMode

  max number of bytes to read in one cycle of Fls_MainFunction (fast mode)
- Fls_LengthType u32MaxReadNormalMode

  max number of bytes to read in one cycle of Fls_MainFunction (normal mode)
- Fls_LengthType u32MaxWriteFastMode

  max number of bytes to write in one cycle of Fls_MainFunction (fast mode)
- Fls_LengthType u32MaxWriteNormalMode

  max number of bytes to write in one cycle of Fls_MainFunction (normal mode)
- Fls_SectorCountType u32SectorCount

  number of configured logical sectors
- const Fls_AddressType(∗ paSectorEndAddr)[ ]

  pointer to array containing last logical address of each configured sector
- const Fls_LengthType(∗ paSectorSize)[ ]

  pointer to array containing sector size of each configured sector
- const Fls_Flash_InternalSectorInfoType ∗const (∗ pSectorList)[ ]

  pointer to array containing physical sector ID of each configured sector
- const uint8(∗ paSectorFlags)[ ]

  pointer to array containing flags set of each configured sector
- const Fls_LengthType(∗ paSectorPageSize)[ ]

  pointer to array containing page size information of each configured sector
- const Fls_HwChType(∗ paHwCh)[ ]

  Pointer to array containing the hardware channel(internal, external_qspi, external_emmc) of each configured sector.
- const uint32(∗ paSectorHwAddress)[ ]

  Pointer to array containing the configured hardware start address of each external sector.
- const Fls_QspiCfgConfigType ∗ pFlsQspiCfgConfig

  Pointer to configuration structure of QSPI.
- const Fls_InternalConfigType ∗ pFlsInternalCfgConfig

  Pointer to configuration structure internal flash.
- Fls_CrcType u16ConfigCrc

  FLS Config Set CRC checksum.

6.1.2.3.1 Field Documentation

6.1.2.3.1.1 acErasePtr `Fls_AcErasePtrType acErasePtr`

pointer to erase access code function in RAM or ROM

Definition at line 387 of file Fls_Types.h.

6.1.2.3.1.2 acWritePtr `Fls_AcWritePtrType acWritePtr`

pointer to write access code function in RAM or ROM

Definition at line 391 of file Fls_Types.h.

6.1.2.3.1.3 acCallBackPtr `Fls_ACCallbackPtrType acCallBackPtr`

pointer to ac callback function

Definition at line 395 of file Fls_Types.h.

6.1.2.3.1.4 jobEndNotificationPtr `Fls_JobEndNotificationPtrType jobEndNotificationPtr`

pointer to job end notification function

Definition at line 399 of file Fls_Types.h.

6.1.2.3.1.5 jobErrorNotificationPtr `Fls_JobErrorNotificationPtrType jobErrorNotificationPtr`

pointer to job error notification function

Definition at line 403 of file Fls_Types.h.

6.1.2.3.1.6 FlsReadFunctionCallout `Fls_ReadFunctionPtrType FlsReadFunctionCallout`

pointer to read to flash memory callout

Definition at line 407 of file Fls_Types.h.

6.1.2.3.1.7 eDefaultMode `MemIf_ModeType eDefaultMode`

default FLS device mode after initialization (MEMIF_MODE_FAST, MEMIF_MODE_SLOW)

Definition at line 411 of file Fls_Types.h.

6.1.2.3.1.8 u32MaxReadFastMode `Fls_LengthType u32MaxReadFastMode`

max number of bytes to read in one cycle of Fls_MainFunction (fast mode)

Definition at line 415 of file Fls_Types.h.

6.1.2.3.1.9 u32MaxReadNormalMode `Fls_LengthType u32MaxReadNormalMode`

max number of bytes to read in one cycle of Fls_MainFunction (normal mode)

Definition at line 419 of file Fls_Types.h.

6.1.2.3.1.10 u32MaxWriteFastMode `Fls_LengthType u32MaxWriteFastMode`

max number of bytes to write in one cycle of Fls_MainFunction (fast mode)

Definition at line 423 of file Fls_Types.h.

6.1.2.3.1.11 u32MaxWriteNormalMode `Fls_LengthType u32MaxWriteNormalMode`

max number of bytes to write in one cycle of Fls_MainFunction (normal mode)

Definition at line 427 of file Fls_Types.h.

6.1.2.3.1.12 u32SectorCount `Fls_SectorCountType u32SectorCount`

number of configured logical sectors

Definition at line 431 of file Fls_Types.h.

6.1.2.3.1.13 paSectorEndAddr `const Fls_AddressType(* paSectorEndAddr)[]`

pointer to array containing last logical address of each configured sector

Definition at line 435 of file Fls_Types.h.

6.1.2.3.1.14 paSectorSize `const Fls_LengthType(* paSectorSize)[]`

pointer to array containing sector size of each configured sector

Definition at line 439 of file Fls_Types.h.

6.1.2.3.1.15 pSectorList `const Fls_Flash_InternalSectorInfoType* const(* pSectorList)[]`

pointer to array containing physical sector ID of each configured sector

Definition at line 443 of file Fls_Types.h.

6.1.2.3.1.16 paSectorFlags `const uint8(* paSectorFlags)[]`

pointer to array containing flags set of each configured sector

Definition at line 447 of file Fls_Types.h.

6.1.2.3.1.17 paSectorPageSize `const Fls_LengthType(* paSectorPageSize)[]`

pointer to array containing page size information of each configured sector

Definition at line 451 of file Fls_Types.h.

6.1.2.3.1.18 paHwCh `const Fls_HwChType(* paHwCh)[]`

Pointer to array containing the hardware channel(internal, external_qspi, external_emmc) of each configured sector.

Definition at line 455 of file Fls_Types.h.

6.1.2.3.1.19 paSectorHwAddress `const uint32(* paSectorHwAddress)[]`

Pointer to array containing the configured hardware start address of each external sector.

Definition at line 459 of file Fls_Types.h.

6.1.2.3.1.20 pFlsQspiCfgConfig `const Fls_QspiCfgConfigType* pFlsQspiCfgConfig`

Pointer to configuration structure of QSPI.

Definition at line 462 of file Fls_Types.h.

6.1.2.3.1.21 pFlsInternalCfgConfig `const Fls_InternalConfigType* pFlsInternalCfgConfig`

Pointer to configuration structure internal flash.

Definition at line 466 of file Fls_Types.h.

6.1.2.3.1.22 u16ConfigCrc `Fls_CrcType u16ConfigCrc`

FLS Config Set CRC checksum.

Definition at line 470 of file Fls_Types.h.

6.1.3 Macro Definition Documentation

6.1.3.1 FLS_DEVICE_INSTANCE_INVALID

```
#define FLS_DEVICE_INSTANCE_INVALID
```

Invalid device instance

Definition at line 136 of file Fls.h.

6.1.3.2 FLS_API_VENDOR_ID

```
#define FLS_API_VENDOR_ID
```

Version Check parameters.

Definition at line 59 of file Fls_Api.h.

6.1.3.3 FLS_MODULE_ID

```
#define FLS_MODULE_ID
```

AUTOSAR module identification.

Definition at line 105 of file Fls_Api.h.

6.1.3.4 FLS_INSTANCE_ID

```
#define FLS_INSTANCE_ID
```

AUTOSAR module instance identification.

Definition at line 109 of file Fls_Api.h.

6.1.3.5 FLS_E_PARAM_CONFIG

```
#define FLS_E_PARAM_CONFIG
```

Development error codes (passed to DET).

API service called with wrong config parameter

Definition at line 117 of file Fls_Api.h.

6.1.3.6 FLS_E_PARAM_ADDRESS

```
#define FLS_E_PARAM_ADDRESS
```

API service called with wrong address parameter.

Definition at line 121 of file Fls_Api.h.

6.1.3.7 FLS_E_PARAM_LENGTH

```
#define FLS_E_PARAM_LENGTH
```

API service called with wrong length parameter.

Definition at line 125 of file Fls_Api.h.

6.1.3.8 FLS_E_PARAM_DATA

```
#define FLS_E_PARAM_DATA
```

API service called with wrong data parameter.

Definition at line 129 of file Fls_Api.h.

6.1.3.9 FLS_E_UNINIT

```
#define FLS_E_UNINIT
```

API service called without module initialization.

Definition at line 133 of file Fls_Api.h.

6.1.3.10 FLS_E_BUSY

```
#define FLS_E_BUSY
```

API service called while driver still busy.

Definition at line 137 of file Fls_Api.h.

6.1.3.11 FLS_E_PARAM_POINTER

```
#define FLS_E_PARAM_POINTER
```

API service called with NULL pointer.

Definition at line 141 of file Fls_Api.h.

6.1.3.12 FLS_E_VERIFY_ERASE_FAILED

```
#define FLS_E_VERIFY_ERASE_FAILED
```

Runtime error codes (passed to DET).

Erase verification (blank check) failed

Definition at line 149 of file Fls_Api.h.

6.1.3.13 FLS_E_VERIFY_WRITE_FAILED

```
#define FLS_E_VERIFY_WRITE_FAILED
```

Write verification (compare) failed.

Definition at line 153 of file Fls_Api.h.

6.1.3.14 FLS_E_TIMEOUT

```
#define FLS_E_TIMEOUT
```

Timeout exceeded.

Definition at line 157 of file Fls_Api.h.

6.1.3.15 FLS_E_ERASE_FAILED

```
#define FLS_E_ERASE_FAILED
```

Transient Faults codes (passed to DET).

Flash erase failed (HW)

Definition at line 165 of file Fls_Api.h.

6.1.3.16 FLS_E_WRITE_FAILED

```
#define FLS_E_WRITE_FAILED
```

Flash write failed (HW)

Definition at line 169 of file Fls_Api.h.

6.1.3.17 FLS_E_READ_FAILED

```
#define FLS_E_READ_FAILED
```

Flash read failed (HW)

Definition at line 173 of file Fls_Api.h.

6.1.3.18 FLS_E_COMPARE_FAILED

```
#define FLS_E_COMPARE_FAILED
```

Flash compare failed (HW)

Definition at line 177 of file Fls_Api.h.

6.1.3.19 FLS_INIT_ID

```
#define FLS_INIT_ID
```

All service IDs (passed to DET).

service ID of function: Fls_Init. (passed to DET)

Definition at line 189 of file Fls_Api.h.

6.1.3.20 FLS_ERASE_ID

```
#define FLS_ERASE_ID
```

service ID of function: Fls_Erase. (passed to DET)

Definition at line 193 of file Fls_Api.h.

6.1.3.21 FLS_WRITE_ID

```
#define FLS_WRITE_ID
```

service ID of function: Fls_Write. (passed to DET)

Definition at line 197 of file Fls_Api.h.

6.1.3.22 FLS_CANCEL_ID

```
#define FLS_CANCEL_ID
```

service ID of function: Fls_Cancel. (passed to DET)

Definition at line 201 of file Fls_Api.h.

6.1.3.23 FLS_GETJOBRESULT_ID

```
#define FLS_GETJOBRESULT_ID
```

service ID of function: Fls_GetJobResult. (passed to DET)

Definition at line 205 of file Fls_Api.h.

6.1.3.24 FLS_MAINFUNCTION_ID

```
#define FLS_MAINFUNCTION_ID
```

service ID of function: Fls_MainFunction. (passed to DET)

Definition at line 209 of file Fls_Api.h.

6.1.3.25 FLS_READ_ID

```
#define FLS_READ_ID
```

service ID of function: Fls_Read. (passed to DET)

Definition at line 213 of file Fls_Api.h.

6.1.3.26 FLS_COMPARE_ID

```
#define FLS_COMPARE_ID
```

service ID of function: Fls_Compare. (passed to DET)

Definition at line 217 of file Fls_Api.h.

6.1.3.27 FLS_SETMODE_ID

```
#define FLS_SETMODE_ID
```

service ID of function: Fls_SetMode. (passed to DET)

Definition at line 221 of file Fls_Api.h.

6.1.3.28 FLS_GETVERSIONINFO_ID

```
#define FLS_GETVERSIONINFO_ID
```

service ID of function: Fls_GetVersionInfo. (passed to DET)

Definition at line 225 of file Fls_Api.h.

6.1.3.29 FLS_BLANK_CHECK_ID

```
#define FLS_BLANK_CHECK_ID
```

service ID of function: Fls_BlankCheck. (passed to DET)

Definition at line 229 of file Fls_Api.h.

6.1.3.30 FLS_SECTOR_ERASE_ASYNCH

```
#define FLS_SECTOR_ERASE_ASYNCH
```

All sector flags.

fls sector erase asynch

Definition at line 238 of file Fls_Api.h.

6.1.3.31 FLS_PAGE_WRITE_ASYNCH

```
#define FLS_PAGE_WRITE_ASYNCH
```

fls page write asynch

Definition at line 242 of file Fls_Api.h.

6.1.3.32 FLS_START_SEC_CODE

```
#define FLS_START_SEC_CODE
```

Start of Fls section CODE.

Definition at line 254 of file Fls_Api.h.

6.1.3.33 FLS_STOP_SEC_CODE

```
#define FLS_STOP_SEC_CODE
```

Stop of Fls section CODE.

Definition at line 550 of file Fls_Api.h.

6.1.3.34 FLS_IPW_CFG_INVALID

```
#define FLS_IPW_CFG_INVALID
```

Invalid configuration, specifies unused device

Definition at line 108 of file Fls_IPW.h.

6.1.4 Types Reference

6.1.4.1 Fls_SectorIndexType

```
typedef uint32 Fls_SectorIndexType
```

Logical sector index.

Definition at line 235 of file Fls_Types.h.

6.1.4.2 Fls_CrcType

```
typedef uint16 Fls_CrcType
```

Fls CRC Type.

CRC computed over config set.

Definition at line 242 of file Fls_Types.h.

6.1.4.3 Fls_AddressType

```
typedef uint32 Fls_AddressType
```

Fls Address Type.

Address offset from the configured flash base address to access a certain flash memory area.

Definition at line 249 of file Fls_Types.h.

6.1.4.4 Fls_LengthType

```
typedef uint32 Fls_LengthType
```

Fls Length Type.

Number of bytes to read,write,erase,compare

Definition at line 255 of file Fls_Types.h.

6.1.4.5 Fls_SectorCountType

```
typedef uint32 Fls_SectorCountType
```

Fls Sector Count Type.

Number of configured sectors

Definition at line 261 of file Fls_Types.h.

6.1.4.6 Fls_BlockNumberOfSectorType

```
typedef uint8 Fls_BlockNumberOfSectorType
```

Fls BLock Count Type.

Block number of sectors type

Definition at line 267 of file Fls_Types.h.

6.1.4.7 Fls_InternalConfigType

```
typedef Ftfc_ConfigType Fls_InternalConfigType
```

Fls Internal Flash Type.

Configuration structure of internal flash.

Definition at line 273 of file Fls_Types.h.

6.1.4.8 Fls_JobEndNotificationPtrType

```
typedef void(* Fls_JobEndNotificationPtrType) (void)
```

Fls Job End Notification Pointer Type.

Pointer type of Fls_JobEndNotification function

Definition at line 280 of file Fls_Types.h.

6.1.4.9 Fls_JobErrorNotificationPtrType

```
typedef void(* Fls_JobErrorNotificationPtrType) (void)
```

Fls Job Error Notification Pointer Type.

Pointer type of Fls_JobErrorNotification function

Definition at line 286 of file Fls_Types.h.

6.1.4.10 Fls_ACCallbackPtrType

```
typedef void(* Fls_ACCallbackPtrType) (void)
```

Pointer type of Fls_AC_Callback function.

Definition at line 292 of file Fls_Types.h.

6.1.4.11 Fls_AcErasePtrType

```
typedef void(* Fls_AcErasePtrType) (void(*CallBack)(void))
```

Define pointer type of erase access code function.

Definition at line 296 of file Fls_Types.h.

6.1.4.12 Fls_AcWritePtrType

```
typedef void(* Fls_AcWritePtrType) (void(*CallBack)(void))
```

Define pointer type of write access code function.

Definition at line 301 of file Fls_Types.h.

6.1.4.13 Fls_ReadFunctionPtrType

```
typedef void(* Fls_ReadFunctionPtrType) (void)
```

Pointer type of Fls_ReadFunctionPtrType function.

The callout for the user to check for ECC errors for Internal Flash memories. In this callout, the user can schedule a task that reads from flash memory to a read source buffer and check/handle for an ECC exception.

Definition at line 312 of file Fls_Types.h.

6.1.5 Enum Reference

6.1.5.1 Fls_HwChType

```
enum Fls_HwChType
```

Flash sector channel type.

Definition at line 141 of file Fls_Types.h.

6.1.5.2 Fls_JobType

```
enum Fls_JobType
```

Type of job currently executed by Fls_MainFunction.

Enumerator

FLS_JOB_ERASE	erase one or more complete flash sectors
FLS_JOB_WRITE	write one or more complete flash pages
FLS_JOB_READ	read one or more bytes from flash memory
FLS_JOB_COMPARE	compare data buffer with content of flash memory
FLS_JOB_BLANK_CHECK	check content of erased flash memory area

Definition at line 150 of file Fls_Types.h.

6.1.5.3 Fls_LLDReturnType

```
enum Fls_LLDReturnType
```

Result of low-level flash operation.

Enumerator

FLASH_E_OK	operation succeeded
FLASH_E_FAILED	operation failed due to hardware error
FLASH_E_BLOCK_INCONSISTENT	data buffer doesn't match with content of flash memory
FLASH_E_PENDING	operation is pending
FLASH_E_PARTITION_ERR	FlexNVM partition ratio error.

Definition at line 177 of file Fls_Types.h.

6.1.5.4 Fls_LLDJobType

```
enum Fls_LLDJobType
```

Type of job currently executed by Fls_LLDMainFunction.

Enumerator

FLASH_JOB_NONE	no job executed by Fls_LLDMainFunction
FLASH_JOB_ERASE	erase one flash sector
FLASH_JOB_ERASE_TEMP	complete erase and start an interleaved erase flash sector
FLASH_JOB_WRITE	write one or more complete flash pages
FLASH_JOB_ERASE_BLANK_CHECK	erase blank check of flash sector

Definition at line 189 of file Fls_Types.h.

6.1.5.5 Fls_CrcDataSizeType

```
enum Fls_CrcDataSizeType
```

Size of data to be processeed by CRC.

Enumerator

FLS_CRC_8_BITS	crc 8 bits
FLS_CRC_16_BITS	crc 16 bits

Definition at line 219 of file Fls_Types.h.

6.1.6 Function Reference

6.1.6.1 Fls_MainFunction()

```
void Fls_MainFunction (
            void  )
```

Performs actual flash read, write, erase and compare jobs.

Bytes number processed per cycle depends by job type (erase, write, read, compare) current FLS module's operating mode (normal, fast) and write, erase Mode of Execution (sync, async).

Precondition

The module has to be initialized.

Note

This function have to be called cyclically by the Basic Software Module; it will do nothing if there aren't pending job.

6.1.6.2 Fls_Init()

```
void Fls_Init (
            const Fls_ConfigType * ConfigPtr )
```

The function initializes Fls module.

The function sets the internal module variables according to given configuration set.

Parameters

in	*ConfigPtr*	Pointer to flash driver configuration set.

Precondition

`ConfigPtr` must not be `NULL_PTR` and the module status must not be `MEMIF_BUSY`.

6.1.6.3 Fls_Write()

```
Std_ReturnType Fls_Write (
            Fls_AddressType TargetAddress,
            const uint8 * SourceAddressPtr,
            Fls_LengthType Length )
```

Write one or more complete flash pages to the flash device.

Starts a write job asynchronously. The actual job is performed by `Fls_MainFunction`.

Parameters

in	*TargetAddress*	Target address in flash memory. This address offset will be added to the flash memory base address.
in	*SourceAddressPtr*	Pointer to source data buffer.
in	*Length*	Number of bytes to write.

Returns

Std_ReturnType

Return values

E_OK	Write command has been accepted.
E_NOT_OK	Write command has not been accepted.

Precondition

The module has to be initialized and not busy.

Postcondition

`Fls_Write` changes module status and some internal variables (`Fls_u32JobSectorIt`, `Fls_u32Job↩ AddrIt`, `Fls_u32JobAddrEnd`, `Fls_pJobDataSrcPtr`, `Fls_eJob`, `Fls_eJobResult`).

6.1.6.4 Fls_Erase()

```
Std_ReturnType Fls_Erase (
            Fls_AddressType TargetAddress,
            Fls_LengthType Length )
```

Erase one or more complete flash sectors.

Starts an erase job asynchronously. The actual job is performed by the `Fls_MainFunction`.

Parameters

in	*TargetAddress*	Target address in flash memory.
in	*Length*	Number of bytes to erase.

Returns

Std_ReturnType

Return values

E_OK	Erase command has been accepted.
E_NOT_OK	Erase command has not been accepted.

Precondition

The module has to be initialized and not busy.

Postcondition

`Fls_Erase` changes module status and some internal variables (`Fls_u32JobSectorIt`, `Fls_u32Job↩ SectorEnd`, `Fls_Job`, `Fls_eJobResult`).

6.1.6.5 Fls_Read()

```
Std_ReturnType Fls_Read (
            Fls_AddressType SourceAddress,
            uint8 * TargetAddressPtr,
            Fls_LengthType Length )
```

Reads from flash memory.

Starts a read job asynchronously. The actual job is performed by `Fls_MainFunction`.

Parameters

in	*SourceAddress*	Source address in flash memory. This address offset will be added to the flash memory base address.
in	*Length*	Number of bytes to read.
out	*TargetAddressPtr*	Pointer to target data buffer.

Returns

Std_ReturnType

Return values

E_OK	Read command has been accepted
E_NOT_OK	Read command has not been accepted

Precondition

The module has to be initialized and not busy.

Postcondition

`Fls_Read` changes module status and some internal variables (`Fls_u32JobSectorIt`, `Fls_u32JobAddr↩ It`, `Fls_u32JobAddrEnd`, `Fls_pJobDataDestPtr`, `Fls_eJob`, `Fls_eJobResult`).

6.1.7 Variable Documentation

6.1.7.1 Fls_u32JobAddrIt

```
Fls_AddressType Fls_u32JobAddrIt [extern]
```

Logical address of data block currently processed by Fls_MainFunction.

6.1.7.2 Fls_u32JobAddrEnd

```
Fls_AddressType Fls_u32JobAddrEnd [extern]
```

Last logical address to be processed by a job.

6.1.7.3 Fls_u32JobSectorIt

```
volatile Fls_SectorIndexType Fls_u32JobSectorIt  [extern]
```

Index of flash sector currently processed by a job.

Used by all types of job

6.1.7.4 Fls_u32JobSectorEnd

```
Fls_SectorIndexType Fls_u32JobSectorEnd  [extern]
```

Index of last flash sector by current job.

Used to check status of all external flash chips before start jobs or is the last sector in Erease job

6.1.7.5 Fls_eLLDJobResult

```
volatile MemIf_JobResultType Fls_eLLDJobResult  [extern]
```

Result of last flash hardware job.

6.1.7.6 Fls_eLLDJob

```
Fls_LLDJobType Fls_eLLDJob  [extern]
```

Type of current flash hardware job - used for asynchronous operating mode.

6.1.7.7 Fls_pConfigPtr

```
const Fls_ConfigType* Fls_pConfigPtr  [extern]
```

Pointer to current flash module configuration set.

6.2 FTFC IP Driver

6.2.1 Detailed Description

Data Structures

- struct Fls_ExceptionDetailsType

  Detailed information on the exception. More...
- struct Ftfc_ConfigType

  Ftfc Configuration Structure. More...

Macros

- #define FLASH_CMD_PROGRAM_PHRASE

  FCCOB commands IDs.
- #define FTFC_WRITE_DOUBLE_WORD

  Program allignment.
- #define FLS_SIZE_1BYTE

  the number of bytes uses to compare (1 byte).
- #define FLS_SIZE_2BYTE

  the number of bytes uses to compare (2 bytes).
- #define FLS_SIZE_4BYTE

  the number of bytes uses to compare (4 bytes).

Types Reference

- typedef uint8 Fls_CompHandlerReturnType

  Return value of Fls handler function.
- typedef void(∗ Ftfc_StartFlashAccessNotifPtrType) (void)

  Fls Start Flash Access Notification Pointer Type.
- typedef void(∗ Ftfc_FinishedFlashAccessNotifPtrType) (void)

  Fls Finished Flash Access Notification Pointer Type.

Enum Reference

- enum Ftfc_Fls_Ip_FlashBlocksNumberType

  Enumeration of Blocks of memory flash .
- enum Ftfc_Fls_Ip_StatusType

  Enumeration of checking status errors or not.

Function Reference

- Ftfc_Fls_Ip_StatusType Ftfc_Fls_Ip_Init (const Ftfc_ConfigType *Ftfc_Fls_Ip_pInitConfig)

  Initializes the FTCF module.
- Ftfc_Fls_Ip_StatusType Ftfc_Fls_Ip_Abort (void)

  Abort a program or erase operation.
- Ftfc_Fls_Ip_StatusType Ftfc_Fls_Ip_Read (Ftfc_Fls_Ip_AddressType u32SrcAddress, uint8 *pDest↩ AddressPtr, Ftfc_Fls_Ip_LengthType u32Length)

  This function fills data to pDestAddressPtr.
- Ftfc_Fls_Ip_StatusType Ftfc_Fls_Ip_Compare (Ftfc_Fls_Ip_AddressType u32SrcAddress, const uint8 *pCompareAddressPtr, Ftfc_Fls_Ip_LengthType u32Length)

  Checks that there is the desired data at the specified address.
- Ftfc_Fls_Ip_FlashBlocksNumberType Ftfc_Fls_Ip_GetBlockNumberFromAddress (uint32 u32Target↩ Address)

  Get block number from target address.
- Ftfc_Fls_Ip_StatusType Ftfc_Fls_Ip_SectorErase (uint32 u32SectorStartAddress)

  Accepts and erases a selected program flash or data flash sector if possible.
- Ftfc_Fls_Ip_StatusType Ftfc_Fls_Ip_SectorEraseStatus (void)

  Checks the status of the hardware erase started by the Ftfc_Fls_Ip_SectorErase function.
- Ftfc_Fls_Ip_StatusType Ftfc_Fls_Ip_Write (uint32 u32DestAddress, const uint8 *pSourceAddressPtr, uint32 u32Length)

  Writes data into the memory array using the main interface. Initiates the hardware write and then exits.
- Ftfc_Fls_Ip_StatusType Ftfc_Fls_Ip_WriteStatus (void)

  Checks the status of the hardware program started by the FTFC_Ip_Write function.

6.2.2 Data Structure Documentation

6.2.2.1 struct Fls_ExceptionDetailsType

Detailed information on the exception.

The following information will be checked by the driver:

- if there is a pending read, compare,
- data_pt matches address currently accessed by pending flash read or flash compare job,
- if the exception syndrome register indicates DSI or MCI reason,

Definition at line 133 of file Ftfc_Fls_Ip_Types.h.

Data Fields

Type	Name	Description
Fls_InstructionAddressType	instruction_pt	pointer to the instruction that generated the ECC
Fls_DataAddressType	data_pt	data address that caused the ECC error
uint32	syndrome_u32	details on the type of exception

6.2.2.2 struct Ftfc_ConfigType

Ftfc Configuration Structure.

Implements : Ftfc_ConfigType_Class

Definition at line 171 of file Ftfc_Fls_Ip_Types.h.

Data Fields

Type	Name	Description
Ftfc_StartFlashAccessNotifPtrType	startFlashAccessNotifPtr	Pointer to start flash access callout
Ftfc_FinishedFlashAccessNotifPtrType	finishedFlashAccessNotifPtr	Pointer to finish flash access callout

6.2.3 Macro Definition Documentation

6.2.3.1 FLASH_CMD_PROGRAM_PHRASE

```
#define FLASH_CMD_PROGRAM_PHRASE
```

FCCOB commands IDs.

Definition at line 80 of file Ftfc_Fls_Ip_Types.h.

6.2.3.2 FTFC_WRITE_DOUBLE_WORD

```
#define FTFC_WRITE_DOUBLE_WORD
```

Program allignment.

Definition at line 86 of file Ftfc_Fls_Ip_Types.h.

6.2.3.3 FLS_SIZE_1BYTE

```
#define FLS_SIZE_1BYTE
```

the number of bytes uses to compare (1 byte).

Definition at line 92 of file Ftfc_Fls_Ip_Types.h.

6.2.3.4 FLS_SIZE_2BYTE

```
#define FLS_SIZE_2BYTE
```

the number of bytes uses to compare (2 bytes).

Definition at line 98 of file Ftfc_Fls_Ip_Types.h.

6.2.3.5 FLS_SIZE_4BYTE

```
#define FLS_SIZE_4BYTE
```

the number of bytes uses to compare (4 bytes).

Definition at line 104 of file Ftfc_Fls_Ip_Types.h.

6.2.4 Types Reference

6.2.4.1 Fls_CompHandlerReturnType

```
typedef uint8 Fls_CompHandlerReturnType
```

Return value of Fls handler function.

Fls_DsiHandler and Fls_MciHandler can return the following value:

- FLS_UNHANDLED Fls driver is not responsable for this situation
- FLS_HANDLED_SKIP Fls driver is responsable for this exception and want to skip this job

Definition at line 117 of file Ftfc_Fls_Ip_Types.h.

6.2.4.2 Ftfc_StartFlashAccessNotifPtrType

```
typedef void(* Ftfc_StartFlashAccessNotifPtrType) (void)
```

Fls Start Flash Access Notification Pointer Type.

Pointer type of Ftfc_StartFlashAccessNotifPtrType function

Definition at line 157 of file Ftfc_Fls_Ip_Types.h.

6.2.4.3 Ftfc_FinishedFlashAccessNotifPtrType

```
typedef void(* Ftfc_FinishedFlashAccessNotifPtrType) (void)
```

Fls Finished Flash Access Notification Pointer Type.

Pointer type of Ftfc_FinishedFlashAccessNotifPtrType function

Definition at line 164 of file Ftfc_Fls_Ip_Types.h.

6.2.5 Enum Reference

6.2.5.1 Ftfc_Fls_Ip_FlashBlocksNumberType

```
enum Ftfc_Fls_Ip_FlashBlocksNumberType
```

Enumeration of Blocks of memory flash .

Enumerator

FLS_CODE_BLOCK_0	code block number 0
FLS_CODE_BLOCK_1	code block number 1
FLS_CODE_BLOCK_2	code block number 2
FLS_DATA_BLOCK	data block
FLS_BLOCK_INVALID	invalid block

Definition at line 143 of file Ftfc_Fls_Ip_Types.h.

6.2.5.2 Ftfc_Fls_Ip_StatusType

```
enum Ftfc_Fls_Ip_StatusType
```

Enumeration of checking status errors or not.

Enumerator

STATUS_FTFC_FLS_IP_SUCCESS	Successful job
STATUS_FTFC_FLS_IP_BUSY	IP is performing an operation
STATUS_FTFC_FLS_IP_ERROR	Error - general code

Enumerator

STATUS_FTFC_FLS_IP_ERROR_TIMEOUT	Error - exceeded timeout
STATUS_FTFC_FLS_IP_ERROR_INPUT_PA↩ RAM	Error - wrong input parameter
STATUS_FTFC_FLS_IP_ERROR_BLANK_C↩ HECK	Error - selected memory area is not erased
STATUS_FTFC_FLS_IP_ERROR_PROGRAM↩ _VERIFY	Error - selected memory area doesn't contain desired value
STATUS_FTFC_FLS_IP_ERROR_USER_TES↩ T_BREAK_SBC	Error - single bit correction
STATUS_FTFC_FLS_IP_ERROR_USER_TES↩ T_BREAK_DBD	Error - double bit detection
STATUS_FTFC_FLS_IP_SECTOR_UNPROTE↩ CTED	Checked sector is unlocked
STATUS_FTFC_FLS_IP_SECTOR_PROTECT↩ ED	Checked sector is locked

Definition at line 180 of file Ftfc_Fls_Ip_Types.h.

6.2.6 Function Reference

6.2.6.1 Ftfc_Fls_Ip_Init()

```
Ftfc_Fls_Ip_StatusType Ftfc_Fls_Ip_Init (
            const Ftfc_ConfigType * Ftfc_Fls_Ip_pInitConfig )
```

Initializes the FTCF module.

This function will initialize ftfc module and clear all error flags.

Parameters

in	*Ftfc_Fls_Ip_pInitConfig*	Pointer to the driver configuration structure.

Returns

Ftfc_Fls_Ip_StatusType

Return values

STATUS_FTFC_FLS_IP_SUCCESS	Initialization is success
STATUS_FTFC_FLS_IP_ERROR_TIMEOUT	Errors Timeout because wait for the Done bit long time

6.2.6.2 Ftfc_Fls_Ip_Abort()

```
Ftfc_Fls_Ip_StatusType Ftfc_Fls_Ip_Abort (
            void  )
```

Abort a program or erase operation.

This function will abort a program or erase operation in user mode and clear all PGM, APGM, ERS, AERS, EHV, AEHV bits in MCR,AMCRS registers

Returns

Ftfc_Fls_Ip_StatusType

Return values

STATUS_FTFC_FLS_IP_SUCCESS	: The operation is successful.
STATUS_FTFC_FLS_IP_ERROR_TIMEOUT	the operation error because wait for the Done bit long time

6.2.6.3 Ftfc_Fls_Ip_Read()

```
Ftfc_Fls_Ip_StatusType Ftfc_Fls_Ip_Read (
            Ftfc_Fls_Ip_AddressType u32SrcAddress,
            uint8 * pDestAddressPtr,
            Ftfc_Fls_Ip_LengthType u32Length )
```

This function fills data to pDestAddressPtr.

This function fills data to pDestAddressPtr with data from the specified address

Parameters

in	*u32SrcAddress*	The start address of the area to be read.
in	*pDestAddressPtr*	Pointer to the destination of the read.
in	*u32Length*	Read size

Returns

Ftfc_Fls_Ip_StatusType

Return values

STATUS_FTFC_FLS_IP_SUCCESS	Read performed successfully.
STATUS_FTFC_FLS_IP_ERROR_INPUT_PARAM	Input parameters are invalid.
STATUS_FTFC_FLS_IP_ERROR	There was an error while reading.

Precondition

The module has to be initialized and not busy.

6.2.6.4 Ftfc_Fls_Ip_Compare()

```
Ftfc_Fls_Ip_StatusType Ftfc_Fls_Ip_Compare (
            Ftfc_Fls_Ip_AddressType u32SrcAddress,
            const uint8 * pCompareAddressPtr,
            Ftfc_Fls_Ip_LengthType u32Length )
```

Checks that there is the desired data at the specified address.

Checks that there is the desired data at the specified address. If the compare is intented to be a blank check, the pSourceAddressPtr should be NULL.

Parameters

in	*u32SrcAddress*	The start address of the area to be checked.
in	*pCompareAddressPtr*	Pointer to the data expected to be read.
in	*u32Length*	Check size

Returns

Ftfc_Fls_Ip_StatusType

Return values

STATUS_FTFC_FLS_IP_SUCCESS	Read performed successfully.
STATUS_FTFC_FLS_IP_ERROR_INPUT_PA↩ RAM	Input parameters are invalid.
STATUS_FTFC_FLS_IP_ERROR	There was an error while reading.
STATUS_FTFC_FLS_IP_ERROR_PROGRAM_↩ VERIFY	The expected data was not found completely at the specified address

Precondition

The module has to be initialized and not busy.

6.2.6.5 Ftfc_Fls_Ip_GetBlockNumberFromAddress()

```
Ftfc_Fls_Ip_FlashBlocksNumberType Ftfc_Fls_Ip_GetBlockNumberFromAddress (
            uint32 u32TargetAddress )
```

Get block number from target address.

Get block number from target address

Parameters

in	*u32TargetAddress*	target address

Returns

Ftfc_Fls_Ip_GetBlockNumberFromAddress

Return values

The	block number which contains the target address.

6.2.6.6 Ftfc_Fls_Ip_SectorErease()

```
Ftfc_Fls_Ip_StatusType Ftfc_Fls_Ip_SectorErase (
              uint32 u32SectorStartAddress )
```

Accepts and erases a selected program flash or data flash sector if possible.

Accepts an erase job over one of the sectors if possible. Starts the high voltage erase and then exits. The status of the hardware erase must be verified by calling asynchronously the Ftfc_Fls_Ip_SectorEraseStatus function. The Ftfc_Fls_Ip_SectorErase function shall cover all the available sectors.

Parameters

in	*u32SectorStartAddress*	The start address of the sector to be erased.

Returns

Ftfc_Fls_Ip_StatusType

Return values

STATUS_FTFC_FLS_IP_SUCCESS	Hardware erase started successfully
STATUS_FTFC_FLS_IP_ERROR_INPUT_PA↩ RAM	The selected sector is out of bound
STATUS_FTFC_FLS_IP_ERROR	There is another job configured or in progress or `The` sector is locked by another core or couldn't be unlocked.
STATUS_FTFC_FLS_IP_ERROR_TIMEOUT	The erase operation exceeded the timeout - Status value available only if the timeout feature is enabled

Precondition

The module has to be initialized.

6.2.6.7 Ftfc_Fls_Ip_SectorEraseStatus()

```
Ftfc_Fls_Ip_StatusType Ftfc_Fls_Ip_SectorEraseStatus (
            void )
```

Checks the status of the hardware erase started by the Ftfc_Fls_Ip_SectorErase function.

Checks the status of the hardware erase started by the Ftfc_Fls_Ip_SectorErase function.

Returns

Ftfc_Fls_Ip_StatusType

Return values

STATUS_FTFC_FLS_IP_SUCCESS	Erase performed successfully
STATUS_Ftfc_Fls_Ip_BUSY	Hardware erase is still in progress
STATUS_FTFC_FLS_IP_ERROR	There was an error during the hardware erase.
STATUS_FTFC_FLS_IP_ERROR_TIMEOUT	The erase operation exceeded the timeout - Status value available only if the timeout feature is enabled.
STATUS_FTFC_FLS_IP_ERROR_BLANK_CH↩ ECK	The sector was not erased correctly - Status value available only if the blank check feature is enabled

Precondition

The module has to be initialized.

6.2.6.8 Ftfc_Fls_Ip_Write()

```
Ftfc_Fls_Ip_StatusType Ftfc_Fls_Ip_Write (
            uint32 u32DestAddress,
            const uint8 * pSourceAddressPtr,
            uint32 u32Length )
```

Writes data into the memory array using the main interface. Initiates the hardware write and then exits.

Writes data into the memory array using the main interface. Initiates the hardware write and then exits. the status of the hardware erase must be verified by calling asynchronously the Ftfc_Fls_Ip_WriteStatus function.

Parameters

in	*u32DestAddress*	The start address of the write, must be aligned with 8 bytes.
in	*pSourceAddressPtr*	Source program buffer address.
in	*u32Length*	Size in bytes of the flash region to be programed, must be aligned with 8 bytes and the maximum value is 128 bytes.

Returns

Ftfc_Fls_Ip_StatusType

Return values

STATUS_FTFC_FLS_IP_SUCCESS	Program performed successfully
STATUS_FTFC_FLS_IP_ERROR_INPUT_PA↩ RAM	The input parameters are invaid.
STATUS_FTFC_FLS_IP_ERROR	There is another job configured or in progress or `The` sector is locked by another core or couldn't be unlocked.
STATUS_FTFC_FLS_IP_ERROR_TIMEOUT	The erase operation exceeded the timeout - Status value available only if the timeout feature is enabled

Precondition

The module has to be initialized.

6.2.6.9 Ftfc_Fls_Ip_WriteStatus()

```
Ftfc_Fls_Ip_StatusType Ftfc_Fls_Ip_WriteStatus (
            void )
```

Checks the status of the hardware program started by the FTFC_Ip_Write function.

Checks the status of the hardware program started by the FTFC_Ip_Write function.

Returns

Ftfc_Fls_Ip_StatusType

Return values

STATUS_FTFC_FLS_IP_SUCCESS	Program performed successfully
STATUS_Ftfc_Fls_Ip_BUSY	Hardware program is still in progress
STATUS_FTFC_FLS_IP_ERROR	There was an error during the hardware program.
STATUS_FTFC_FLS_IP_ERROR_TIMEOUT	The program operation exceeded the timeout - Status value available only if the timeout feature is enabled.
STATUS_FTFC_FLS_IP_ERROR_PROGRAM_↩ VERIFY	The data was not written corectly into the memory - Status available only of program verify feature is

Precondition

The module has to be initialized.

6.3 FTFC IP Driver

6.3.1 Detailed Description

Function Reference

- void Ftfc_Fls_Ip_InvalidPrefetchBuff_Ram (void)
  Invalidate prefetch buffer before reading to make sure that the driver always reads the new data from flash.

6.3.2 Function Reference

6.3.2.1 Ftfc_Fls_Ip_InvalidPrefetchBuff_Ram()

```
void Ftfc_Fls_Ip_InvalidPrefetchBuff_Ram (
            void  )
```

Invalidate prefetch buffer before reading to make sure that the driver always reads the new data from flash.

Parameters

in	*void*	

Return values

none	

6.4 QSPI IPV Driver

6.4.1 Detailed Description

Data Structures

- struct Qspi_Ip_ControllerAhbConfigType

  AHB configuration structure. More...
- struct Qspi_Ip_ControllerConfigType

  Driver configuration structure. More...
- struct Qspi_Ip_StatusConfigType

  Status register configuration structure. More...
- struct Qspi_Ip_EraseVarConfigType

  Describes one type of erase. More...
- struct Qspi_Ip_EraseConfigType

  Erase capabilities configuration structure. More...
- struct Qspi_Ip_ReadIdConfigType

  Read Id capabilities configuration structure. More...
- struct Qspi_Ip_SuspendConfigType

  Suspend capabilities configuration structure. More...
- struct Qspi_Ip_ResetConfigType

  Soft Reset capabilities configuration structure. More...
- struct Qspi_Ip_LutConfigType

  List of LUT sequences. More...
- struct Qspi_Ip_InitOperationType

  Initialization operation. More...
- struct Qspi_Ip_InitConfigType

  Initialization sequence. More...
- struct Qspi_Ip_MemoryConfigType

  Driver configuration structure. More...
- struct Qspi_Ip_MemoryConnectionType

  Flash-controller conections configuration structure. More...

Macros

- #define QSPI_IP_MAX_READ_SIZE
- #define QSPI_IP_MAX_WRITE_SIZE
- #define QSPI_IP_ERASE_TYPES
- #define QSPI_IP_AHB_BUFFERS

  Number of AHB buffers in the device.
- #define QSPI_IP_LUT_INVALID
- #define QSPI_IP_LUT_SEQ_END

Types Reference

- typedef uint16 Qspi_Ip_InstrOpType

  Operation in a LUT sequence.
- typedef Qspi_Ip_StatusType(∗ Qspi_Ip_InitCalloutPtrType) (uint32 instance)

  Init callout pointer type.
- typedef Qspi_Ip_StatusType(∗ Qspi_Ip_ResetCalloutPtrType) (uint32 instance)

  Reset callout pointer type.
- typedef Qspi_Ip_StatusType(∗ Qspi_Ip_ErrorCheckCalloutPtrType) (uint32 instance)

  Error Check callout pointer type.
- typedef Qspi_Ip_StatusType(∗ Qspi_Ip_EccCheckCalloutPtrType) (uint32 instance, uint32 startAddress, uint32 dataLength)

  Ecc Check callout pointer type.

Enum Reference

- enum Qspi_Ip_StatusType

  qspi return codes
- enum Qspi_Ip_ConnectionType

  flash connection to the QSPI module
- enum Qspi_Ip_OpType

  flash operation type
- enum Qspi_Ip_LutCommandsType

  Lut commands.
- enum Qspi_Ip_LutPadsType

  Lut pad options.
- enum Qspi_Ip_ReadModeType

  Read mode.
- enum Qspi_Ip_DataRateType

  Clock phase used for sampling Rx data.
- enum Qspi_Ip_SampleDelayType

  Delay used for sampling Rx data.
- enum Qspi_Ip_SamplePhaseType

  Clock phase used for sampling Rx data.
- enum Qspi_Ip_FlashDataAlignType

  Alignment of outgoing data with serial clock.

Function Reference

- Qspi_Ip_StatusType Qspi_Ip_Init (uint32 instance, const Qspi_Ip_MemoryConfigType ∗pConfig, const Qspi_Ip_MemoryConnectionType ∗pConnect)

  Initializes the serial flash memory driver.
- Qspi_Ip_StatusType Qspi_Ip_Deinit (uint32 instance)

  De-initializes the serial flash memory driver.
- Qspi_Ip_StatusType Qspi_Ip_EraseBlock (uint32 instance, uint32 address, uint32 size)

Erase a sector in the serial flash.
- Qspi_Ip_StatusType Qspi_Ip_EraseChip (uint32 instance)

  Erase the entire serial flash.
- Qspi_Ip_StatusType Qspi_Ip_GetMemoryStatus (uint32 instance)

  Check the status of the flash device.
- Qspi_Ip_StatusType Qspi_Ip_SetProtection (uint32 instance, uint8 value)

  Sets the protection bits to the requested value.
- Qspi_Ip_StatusType Qspi_Ip_GetProtection (uint32 instance, uint8 ∗value)

  Returns the current value of the protection bits.
- Qspi_Ip_StatusType Qspi_Ip_Reset (uint32 instance)

  Resets the flash device.
- Qspi_Ip_StatusType Qspi_Ip_Enter0XX (uint32 instance)

  Enters 0-X-X (no command) mode. This mode assumes only reads are performed.
- Qspi_Ip_StatusType Qspi_Ip_Exit0XX (uint32 instance)

  Exits 0-X-X (no command) mode. This allows operations other than reads to be performed.
- Qspi_Ip_StatusType Qspi_Ip_ProgramSuspend (uint32 instance)

  Suspends a program operation.
- Qspi_Ip_StatusType Qspi_Ip_ProgramResume (uint32 instance)

  Resumes a program operation.
- Qspi_Ip_StatusType Qspi_Ip_EraseSuspend (uint32 instance)

  Suspends an erase operation.
- Qspi_Ip_StatusType Qspi_Ip_EraseResume (uint32 instance)

  Resumes an erase operation.
- Qspi_Ip_StatusType Qspi_Ip_Read (uint32 instance, uint32 address, uint8 ∗data, uint32 size)

  Read data from serial flash.
- Qspi_Ip_StatusType Qspi_Ip_ReadId (uint32 instance, uint8 ∗data)

  Read manufacturer ID/device ID from serial flash.
- Qspi_Ip_StatusType Qspi_Ip_ProgramVerify (uint32 instance, uint32 address, const uint8 ∗data, uint32 size)

  Verifies the correctness of the programmed data.
- Qspi_Ip_StatusType Qspi_Ip_EraseVerify (uint32 instance, uint32 address, uint32 size)

  Checks whether or not an area in the serial flash is erased.
- Qspi_Ip_StatusType Qspi_Ip_Program (uint32 instance, uint32 address, const uint8 ∗data, uint32 size)

  Writes data in serial flash.
- Qspi_Ip_StatusType Qspi_Ip_RunCommand (uint32 instance, uint16 lut, uint32 addr)

  Launches a simple command for the serial flash.
- Qspi_Ip_StatusType Qspi_Ip_RunReadCommand (uint32 instance, uint16 lut, uint32 addr, uint8 ∗dataRead, const uint8 ∗dataCmp, uint32 size)

  Launches a read command for the serial flash.
- Qspi_Ip_StatusType Qspi_Ip_RunWriteCommand (uint32 instance, uint16 lut, uint32 addr, const uint8 ∗data, uint32 size)

  Launches a write command for the serial flash.
- Qspi_Ip_StatusType Qspi_Ip_AhbReadEnable (uint32 instance)

  Sets up AHB reads to the serial flash.
- Qspi_Ip_StatusType Qspi_Ip_ControllerGetStatus (uint32 instance)

  Check the status of the QSPI controller.
- Qspi_Ip_StatusType Qspi_Ip_ControllerInit (uint32 instance, const Qspi_Ip_ControllerConfigType ∗user←ConfigPtr)

Initializes the qspi driver.
- Qspi_Ip_StatusType Qspi_Ip_ControllerDeinit (uint32 instance)

  De-initialize the qspi driver.
- Qspi_Ip_StatusType Qspi_Ip_ReadSfdp (Qspi_Ip_MemoryConfigType ∗pConfig, const Qspi_Ip_MemoryConnectionTy ∗pConnect)

  Initializes the serial flash memory configuration from SFDP table.

6.4.2 Data Structure Documentation

6.4.2.1 struct Qspi_Ip_ControllerAhbConfigType

AHB configuration structure.

This structure is used to provide configuration parameters for AHB access to the external flash

Definition at line 265 of file Qspi_Ip_Types.h.

Data Fields

Type	Name	Description
uint8	masters[4U]	List of AHB masters assigned to each buffer
uint16	sizes[4U]	List of buffer sizes
boolean	allMasters	Indicates that any master may access the last buffer

6.4.2.2 struct Qspi_Ip_ControllerConfigType

Driver configuration structure.

This structure is used to provide configuration parameters for the qspi driver at initialization time.

Definition at line 296 of file Qspi_Ip_Types.h.

Data Fields

Type	Name	Description
Qspi_Ip_DataRateType	dataRate	Single/double data rate
uint32	memSizeA1	Size of serial flash A1
uint32	memSizeA2	Size of serial flash A2
uint32	memSizeB1	Size of serial flash B1
uint32	memSizeB2	Size of serial flash B2

Data Fields

Type	Name	Description
uint8	csHoldTime	CS hold time, expressed in serial clock cycles
uint8	csSetupTime	CS setup time, expressed in serial clock cycles
uint8	columnAddr	Width of the column address, 0 if not used
boolean	wordAddresable	True if serial flash is word addressable
Qspi_Ip_ReadModeType	readModeA	Read mode for incoming data from serial flash A
Qspi_Ip_ReadModeType	readModeB	Read mode for incoming data from serial flash B
Qspi_Ip_SampleDelayType	sampleDelay	Delay (in clock cycles) used for sampling Rx data
Qspi_Ip_SamplePhaseType	samplePhase	Clock phase used for sampling Rx data
boolean	dqsLatency	Enable DQS latency for reads (Hyperflash)
Qspi_Ip_FlashDataAlignType	dataAlign	Alignment of output data sent to serial flash
uint8	io2IdleValueA	(0 / 1) Logic level of IO[2] signal when not used on side A
uint8	io3IdleValueA	(0 / 1) Logic level of IO[3] signal when not used on side A
uint8	io2IdleValueB	(0 / 1) Logic level of IO[2] signal when not used on side B
uint8	io3IdleValueB	(0 / 1) Logic level of IO[3] signal when not used on side B
Qspi_Ip_ControllerAhbConfigType	ahbConfig	AHB buffers configuration

6.4.2.3 struct Qspi_Ip_StatusConfigType

Status register configuration structure.

This structure contains information about the status registers of the external flash

Definition at line 337 of file Qspi_Ip_Types.h.

Data Fields

Type	Name	Description
uint16	statusRegInitReadLut	Command used to read the status register during initialization
uint16	statusRegReadLut	Command used to read the status register
uint16	statusRegWriteLut	Command used to write the status register
uint16	writeEnableSRLut	Write enable command used before writing to status register
uint16	writeEnableLut	Write enable command used before write or erase operations
uint8	regSize	Size in bytes of status register
uint8	busyOffset	Position of "busy" bit inside status register
uint8	busyValue	Value of "busy" bit which indicates that the device is busy; can be 0 or 1
uint8	writeEnableOffset	Position of "write enable" bit inside status register
uint8	blockProtectionOffset	Offset of block protection bits inside status register
uint8	blockProtectionWidth	Width of block protection bitfield
uint8	blockProtectionValue	Value of block protection bitfield, indicate the protected area

6.4.2.4 struct Qspi__Ip__EraseVarConfigType

Describes one type of erase.

This structure contains information about one type of erase supported by the external flash

Definition at line 359 of file Qspi__Ip__Types.h.

Data Fields

Type	Name	Description
uint16	eraseLut	Lut index for erase command
uint8	size	Size of the erased area: 2 ^ size; e.g. 0x0C means 4 Kbytes

6.4.2.5 struct Qspi__Ip__EraseConfigType

Erase capabilities configuration structure.

This structure contains information about the erase capabilities of the external flash

Definition at line 371 of file Qspi_Ip_Types.h.

Data Fields

Type	Name	Description
Qspi_Ip_EraseVarConfigType	eraseTypes[4U]	Erase types supported by the device
uint16	chipEraseLut	Lut index for chip erase command

6.4.2.6 struct Qspi_Ip_ReadIdConfigType

Read Id capabilities configuration structure.

This structure contains information about the read manufacturer/device ID command

Definition at line 383 of file Qspi_Ip_Types.h.

Data Fields

Type	Name	Description
uint16	readIdLut	Read Id command
uint16	readIdSize	Size of data returned by Read Id command
uint32	readIdExpected	Device ID configured value (Memory density | Memory type | Manufacturer ID)

6.4.2.7 struct Qspi_Ip_SuspendConfigType

Suspend capabilities configuration structure.

This structure contains information about the Program / Erase Suspend capabilities of the external flash

Definition at line 396 of file Qspi_Ip_Types.h.

Data Fields

Type	Name	Description
uint16	eraseSuspendLut	Lut index for the erase suspend operation
uint16	eraseResumeLut	Lut index for the erase resume operation
uint16	programSuspendLut	Lut index for the program suspend operation
uint16	programResumeLut	Lut index for the program resume operation

6.4.2.8 struct Qspi_Ip_ResetConfigType

Soft Reset capabilities configuration structure.

This structure contains information about the Soft Reset capabilities of the external flash

Definition at line 410 of file Qspi_Ip_Types.h.

Data Fields

Type	Name	Description
uint16	resetCmdLut	First command in reset sequence
uint8	resetCmdCount	Number of commands in reset sequence

6.4.2.9 struct Qspi_Ip_LutConfigType

List of LUT sequences.

List of LUT sequences. Each sequence describes a command to the external flash. Sequences are separated by a 0 operation

Definition at line 422 of file Qspi_Ip_Types.h.

Data Fields

Type	Name	Description
uint16	opCount	Number of operations in the LUT table
Qspi_Ip_InstrOpType ∗	lutOps	List of operations

6.4.2.10 struct Qspi_Ip_InitOperationType

Initialization operation.

This structure describes one initialization operation.

Definition at line 434 of file Qspi_Ip_Types.h.

Data Fields

Type	Name	Description
Qspi_Ip_OpType	opType	Operation type

Data Fields

Type	Name	Description
uint16	command1Lut	Index of first command sequence in Lut; for RMW type this is the read command
uint16	command2Lut	Index of second command sequence in Lut, only used for RMW type, this is the write command
uint16	weLut	Index of write enable sequence in Lut, only used for Write and RMW type
uint32	addr	Address, if used in command.
uint8	size	Size in bytes of configuration register
uint8	shift	Position of configuration field inside the register
uint8	width	Width in bits of configuration field.
uint32	value	Value to set in the field
const Qspi_Ip_ControllerConfigType *	ctrlCfgPtr	New controller configuration, valid only for QSPI_IP_OP_TYPE_QSPI_CFG type

6.4.2.11 struct Qspi_Ip_InitConfigType

Initialization sequence.

Describe sequence that will be performed only once during initialization to put the flash in the desired state for operation. This may include, for example, setting the QE bit, activating 4-byte addressing, activating XPI mode

Definition at line 455 of file Qspi_Ip_Types.h.

Data Fields

Type	Name	Description
uint8	opCount	Number of operations
Qspi_Ip_InitOperationType *	operations	List of operations

6.4.2.12 struct Qspi_Ip_MemoryConfigType

Driver configuration structure.

This structure is used to provide configuration parameters for the external flash driver at initialization time.

Definition at line 469 of file Qspi_Ip_Types.h.

Data Fields

Type	Name	Description
uint32	memSize	Memory size (in bytes)
uint32	pageSize	Page size (in bytes)
uint16	readLut	Command used to read data from flash
uint16	writeLut	Command used to write data to flash
uint16	read0xxLut	0-x-x mode read command
uint16	read0xxLutAHB	0-x-x mode AHB read command
Qspi_Ip_ReadIdConfigType	readIdSettings	Erase settings of the external flash
Qspi_Ip_EraseConfigType	eraseSettings	Erase settings of the external flash
Qspi_Ip_StatusConfigType	statusConfig	Status register information
Qspi_Ip_SuspendConfigType	suspendSettings	Program / Erase Suspend settings
Qspi_Ip_ResetConfigType	resetSettings	Soft Reset settings, used at runtime
Qspi_Ip_ResetConfigType	initResetSettings	Soft Reset settings, used for first time reset
Qspi_Ip_InitConfigType	initConfiguration	Operations for initial flash configuration
Qspi_Ip_LutConfigType	lutSequences	List of LUT sequences describing flash commands
Qspi_Ip_InitCalloutPtrType	initCallout	Pointer to init callout
Qspi_Ip_ResetCalloutPtrType	resetCallout	Pointer to reset callout
Qspi_Ip_ErrorCheckCalloutPtrType	errorCheckCallout	Pointer to error check callout
Qspi_Ip_EccCheckCalloutPtrType	eccCheckCallout	Pointer to ecc check callout
const Qspi_Ip_ControllerConfigType ∗	ctrlAutoCfgPtr	Initial controller configuration, if needed

6.4.2.13 struct Qspi_Ip_MemoryConnectionType

Flash-controller conections configuration structure.

This structure specifies thte connecctions of each flash device to QSPI controllers at initialization time.

Definition at line 499 of file Qspi_Ip_Types.h.

Data Fields

Type	Name	Description
uint32	qspiInstance	QSPI Instance where this device is connected
Qspi_Ip_ConnectionType	connectionType	Device connection to QSPI module
uint8	memAlignment	Memory alignment required by the external flash

6.4.3 Macro Definition Documentation

6.4.3.1 QSPI_IP_MAX_READ_SIZE

```
#define QSPI_IP_MAX_READ_SIZE
```

Maximum number of bytes then can be read in one operation

Definition at line 111 of file Qspi_Ip.h.

6.4.3.2 QSPI_IP_MAX_WRITE_SIZE

```
#define QSPI_IP_MAX_WRITE_SIZE
```

Maximum number of bytes then can be written in one operation

Definition at line 113 of file Qspi_Ip.h.

6.4.3.3 QSPI_IP_ERASE_TYPES

```
#define QSPI_IP_ERASE_TYPES
```

Number of erase types that can be supported by a flash device

Definition at line 92 of file Qspi_Ip_Types.h.

6.4.3.4 QSPI_IP_AHB_BUFFERS

```
#define QSPI_IP_AHB_BUFFERS
```

Number of AHB buffers in the device.

Definition at line 95 of file Qspi_Ip_Types.h.

6.4.3.5 QSPI_IP_LUT_INVALID

```
#define QSPI_IP_LUT_INVALID
```

Invalid sequence number in virtual LUT, used for unsupported features

Definition at line 98 of file Qspi_Ip_Types.h.

6.4.3.6 QSPI_IP_LUT_SEQ_END

```
#define QSPI_IP_LUT_SEQ_END
```

End operation for a LUT sequence

Definition at line 100 of file Qspi_Ip_Types.h.

6.4.4 Types Reference

6.4.4.1 Qspi_Ip_InstrOpType

```
typedef uint16 Qspi_Ip_InstrOpType
```

Operation in a LUT sequence.

This type describes one basic operation inside a LUT sequence. Each operation contains:

- instruction (6 bits)
- number of PADs (2 bits)
- operand (8 bits) Qspi_Ip_LutCommandsType and Qspi_Ip_LutPadsType types should be used to form operations

Definition at line 183 of file Qspi_Ip_Types.h.

6.4.4.2 Qspi_Ip_InitCalloutPtrType

```
typedef Qspi_Ip_StatusType(* Qspi_Ip_InitCalloutPtrType) (uint32 instance)
```

Init callout pointer type.

Definition at line 239 of file Qspi_Ip_Types.h.

6.4.4.3 Qspi_Ip_ResetCalloutPtrType

```
typedef Qspi_Ip_StatusType(* Qspi_Ip_ResetCalloutPtrType) (uint32 instance)
```

Reset callout pointer type.

Definition at line 243 of file Qspi_Ip_Types.h.

6.4.4.4 Qspi_Ip_ErrorCheckCalloutPtrType

```
typedef Qspi_Ip_StatusType(* Qspi_Ip_ErrorCheckCalloutPtrType) (uint32 instance)
```

Error Check callout pointer type.

Definition at line 247 of file Qspi_Ip_Types.h.

6.4.4.5 Qspi_Ip_EccCheckCalloutPtrType

```
typedef Qspi_Ip_StatusType(* Qspi_Ip_EccCheckCalloutPtrType) (uint32 instance, uint32 startAddress, uint32
dataLength)
```

Ecc Check callout pointer type.

Definition at line 251 of file Qspi_Ip_Types.h.

6.4.5 Enum Reference

6.4.5.1 Qspi_Ip_StatusType

```
enum Qspi_Ip_StatusType
```

qspi return codes

Enumerator

STATUS_QSPI_IP_SUCCESS	Successful job
STATUS_QSPI_IP_ERROR	IP is performing an operation
STATUS_QSPI_IP_BUSY	Error - general code
STATUS_QSPI_IP_TIMEOUT	Error - exceeded timeout
STATUS_QSPI_IP_ERROR_PROGRAM_VERI↩ FY	Error - selected memory area doesn't contain desired value

Definition at line 105 of file Qspi_Ip_Types.h.

6.4.5.2 Qspi_Ip_ConnectionType

`enum Qspi_Ip_ConnectionType`

flash connection to the QSPI module

Enumerator

QSPI_IP_SIDE_A1	Serial flash connected on side A1
QSPI_IP_SIDE_A2	Serial flash connected on side A2
QSPI_IP_SIDE_B1	Serial flash connected on side B1
QSPI_IP_SIDE_B2	Serial flash connected on side B2

Definition at line 117 of file Qspi_Ip_Types.h.

6.4.5.3 Qspi_Ip_OpType

`enum Qspi_Ip_OpType`

flash operation type

Enumerator

QSPI_IP_OP_TYPE_CMD	Simple command
QSPI_IP_OP_TYPE_WRITE_REG	Write value in external flash register

Enumerator

QSPI_IP_OP_TYPE_RMW_REG	RMW command on external flash register
QSPI_IP_OP_TYPE_READ_REG	Read external flash register until expected value is read
QSPI_IP_OP_TYPE_QSPI_CFG	Re-configure QSPI controller

Definition at line 128 of file Qspi_Ip_Types.h.

6.4.5.4 Qspi_Ip_LutCommandsType

```
enum Qspi_Ip_LutCommandsType
```

Lut commands.

Enumerator

QSPI_IP_LUT_INSTR_STOP	End of sequence
QSPI_IP_LUT_INSTR_CMD	Command
QSPI_IP_LUT_INSTR_ADDR	Address
QSPI_IP_LUT_INSTR_DUMMY	Dummy cycles
QSPI_IP_LUT_INSTR_MODE	8-bit mode
QSPI_IP_LUT_INSTR_MODE2	2-bit mode
QSPI_IP_LUT_INSTR_MODE4	4-bit mode
QSPI_IP_LUT_INSTR_READ	Read data
QSPI_IP_LUT_INSTR_WRITE	Write data
QSPI_IP_LUT_INSTR_JMP_ON_CS	Jump on chip select deassert and stop
QSPI_IP_LUT_INSTR_ADDR_DDR	Address - DDR mode
QSPI_IP_LUT_INSTR_MODE_DDR	8-bit mode - DDR mode
QSPI_IP_LUT_INSTR_MODE2_DDR	2-bit mode - DDR mode
QSPI_IP_LUT_INSTR_MODE4_DDR	4-bit mode - DDR mode
QSPI_IP_LUT_INSTR_READ_DDR	Read data - DDR mode

Enumerator

QSPI_IP_LUT_INSTR_WRITE_DDR	Write data - DDR mode
QSPI_IP_LUT_INSTR_DATA_LEARN	Data learning pattern
QSPI_IP_LUT_INSTR_CMD_DDR	Command - DDR mode
QSPI_IP_LUT_INSTR_CADDR	Column address
QSPI_IP_LUT_INSTR_CADDR_DDR	Column address - DDR mode
QSPI_IP_LUT_INSTR_JMP_TO_SEQ	Jump on chip select deassert and continue

Definition at line 139 of file Qspi_Ip_Types.h.

6.4.5.5 Qspi_Ip_LutPadsType

```
enum Qspi_Ip_LutPadsType
```

Lut pad options.

Enumerator

QSPI_IP_LUT_PADS↩ _1	1 Pad
QSPI_IP_LUT_PADS↩ _2	2 Pads
QSPI_IP_LUT_PADS↩ _4	4 Pads
QSPI_IP_LUT_PADS↩ _8	8 Pads

Definition at line 166 of file Qspi_Ip_Types.h.

6.4.5.6 Qspi_Ip_ReadModeType

```
enum Qspi_Ip_ReadModeType
```

Read mode.

Enumerator

QSPI_IP_READ_MODE_EXTERNAL_DQS	Use external strobe signal

Definition at line 187 of file Qspi_Ip_Types.h.

6.4.5.7 Qspi_Ip_DataRateType

```
enum Qspi_Ip_DataRateType
```

Clock phase used for sampling Rx data.

Enumerator

QSPI_IP_DATA_RATE_SDR	Single data rate
QSPI_IP_DATA_RATE_DDR	Double data rate

Definition at line 204 of file Qspi_Ip_Types.h.

6.4.5.8 Qspi_Ip_SampleDelayType

```
enum Qspi_Ip_SampleDelayType
```

Delay used for sampling Rx data.

Enumerator

QSPI_IP_SAMPLE_DELAY_SAME_DQS	Same DQS
QSPI_IP_SAMPLE_DELAY_HALFCYCLE_EARLY_DQS	Half-cycle early DQS

Definition at line 213 of file Qspi_Ip_Types.h.

6.4.5.9 Qspi_Ip_SamplePhaseType

```
enum Qspi_Ip_SamplePhaseType
```

Clock phase used for sampling Rx data.

Enumerator

QSPI_IP_SAMPLE_PHASE_NON_INVERTED	Sampling at non-inverted clock
QSPI_IP_SAMPLE_PHASE_INVERTED	Sampling at inverted clock

Definition at line 221 of file Qspi_Ip_Types.h.

6.4.5.10 Qspi_Ip_FlashDataAlignType

```
enum Qspi_Ip_FlashDataAlignType
```

Alignment of outgoing data with serial clock.

Enumerator

QSPI_IP_FLASH_DATA_ALIGN_REFCLK	Data aligned with the posedge of Internal reference clock of QSPI
QSPI_IP_FLASH_DATA_ALIGN_2X_REFCLK	Data aligned with 2x serial flash half clock

Definition at line 229 of file Qspi_Ip_Types.h.

6.4.6 Function Reference

6.4.6.1 Qspi_Ip_Init()

```
Qspi_Ip_StatusType Qspi_Ip_Init (
            uint32 instance,
            const Qspi_Ip_MemoryConfigType * pConfig,
            const Qspi_Ip_MemoryConnectionType * pConnect )
```

Initializes the serial flash memory driver.

This function initializes the external flash driver and prepares it for operation.

Parameters

instance	External flash instance number
pConfig	Pointer to the driver configuration structure.
pConnect	Pointer to the flash device connection structure.

Returns

Error or success status returned by API

6.4.6.2 Qspi_Ip_Deinit()

```
Qspi_Ip_StatusType Qspi_Ip_Deinit (
            uint32 instance )
```

De-initializes the serial flash memory driver.

This function de-initializes the qspi driver. The driver can't be used again until reinitialized. The state structure is no longer needed by the driver and may be freed after calling this function.

Parameters

instance	External flash instance number

Returns

Error or success status returned by API

6.4.6.3 Qspi_Ip_EraseBlock()

```
Qspi_Ip_StatusType Qspi_Ip_EraseBlock (
            uint32 instance,
            uint32 address,
            uint32 size )
```

Erase a sector in the serial flash.

This function performs one erase sector (block) operation on the external flash. The erase size must match one of the device's erase types.

Parameters

instance	External flash instance number
address	Address of sector to be erased
size	Size of the sector to be erase. The sector size must match one of the supported erase sizes of the device.

Returns

Error or success status returned by API

6.4.6.4 Qspi_Ip_EraseChip()

```
Qspi_Ip_StatusType Qspi_Ip_EraseChip (
            uint32 instance )
```

Erase the entire serial flash.

Parameters

instance	External flash instance number

Returns

Error or success status returned by API

6.4.6.5 Qspi_Ip_GetMemoryStatus()

```
Qspi_Ip_StatusType Qspi_Ip_GetMemoryStatus (
            uint32 instance )
```

Check the status of the flash device.

Parameters

instance	External flash instance number

Returns

Error or success status returned by API

6.4.6.6 Qspi_Ip_SetProtection()

```
Qspi_Ip_StatusType Qspi_Ip_SetProtection (
            uint32 instance,
            uint8 value )
```

Sets the protection bits to the requested value.

Parameters

instance	External flash instance number
value	New value for the protection bits

Returns

Error or success status returned by API

6.4.6.7 Qspi_Ip_GetProtection()

```
Qspi_Ip_StatusType Qspi_Ip_GetProtection (
            uint32 instance,
            uint8 * value )
```

Returns the current value of the protection bits.

Parameters

instance	External flash instance number
value	Current value of the protection bits

Returns

Error or success status returned by API

6.4.6.8 Qspi_Ip_Reset()

```
Qspi_Ip_StatusType Qspi_Ip_Reset (
            uint32 instance )
```

Resets the flash device.

Parameters

instance	External flash instance number

Returns

Error or success status returned by API

6.4.6.9 Qspi_Ip_Enter0XX()

```
Qspi_Ip_StatusType Qspi_Ip_Enter0XX (
            uint32 instance )
```

Enters 0-X-X (no command) mode. This mode assumes only reads are performed.

Parameters

instance	External flash instance number

Returns

Error or success status returned by API

6.4.6.10 Qspi_Ip_Exit0XX()

```
Qspi_Ip_StatusType Qspi_Ip_Exit0XX (
            uint32 instance )
```

Exits 0-X-X (no command) mode. This allows operations other than reads to be performed.

Parameters

instance	External flash instance number

Returns

Error or success status returned by API

6.4.6.11 Qspi_Ip_ProgramSuspend()

```
Qspi_Ip_StatusType Qspi_Ip_ProgramSuspend (
            uint32 instance )
```

Suspends a program operation.

Parameters

instance	External flash instance number

Returns

Error or success status returned by API

6.4.6.12 Qspi_Ip_ProgramResume()

```
Qspi_Ip_StatusType Qspi_Ip_ProgramResume (
             uint32 instance )
```

Resumes a program operation.

Parameters

instance	External flash instance number

Returns

Error or success status returned by API

6.4.6.13 Qspi_Ip_EraseSuspend()

```
Qspi_Ip_StatusType Qspi_Ip_EraseSuspend (
             uint32 instance )
```

Suspends an erase operation.

Parameters

instance	External flash instance number

Returns

Error or success status returned by API

6.4.6.14 Qspi_Ip_EraseResume()

```
Qspi_Ip_StatusType Qspi_Ip_EraseResume (
            uint32 instance )
```

Resumes an erase operation.

Parameters

instance	External flash instance number

Returns

Error or success status returned by API

6.4.6.15 Qspi_Ip_Read()

```
Qspi_Ip_StatusType Qspi_Ip_Read (
            uint32 instance,
            uint32 address,
            uint8 * data,
            uint32 size )
```

Read data from serial flash.

Parameters

instance	External flash instance number
address	Start address for read operation
data	Buffer where to store read data
size	Size of data buffer

Returns

Error or success status returned by API

6.4.6.16 Qspi_Ip_ReadId()

```
Qspi_Ip_StatusType Qspi_Ip_ReadId (
            uint32 instance,
            uint8 * data )
```

Read manufacturer ID/device ID from serial flash.

Parameters

instance	External flash instance number
data	Buffer where to store read data. Buffer size must match ReadId initialization settings.

Returns

Error or success status returned by API

6.4.6.17 Qspi_Ip_ProgramVerify()

```
Qspi_Ip_StatusType Qspi_Ip_ProgramVerify (
            uint32 instance,
            uint32 address,
            const uint8 * data,
            uint32 size )
```

Verifies the correctness of the programmed data.

Parameters

instance	External flash instance number
address	Start address of area to be verified
data	Data to be verified
size	Size of area to be verified

Returns

Error or success status returned by API

6.4.6.18 Qspi_Ip_EraseVerify()

```
Qspi_Ip_StatusType Qspi_Ip_EraseVerify (
            uint32 instance,
            uint32 address,
            uint32 size )
```

Checks whether or not an area in the serial flash is erased.

Parameters

instance	External flash instance number
address	Start address of area to be verified
size	Size of area to be verified

Returns

Error or success status returned by API

6.4.6.19 Qspi_Ip_Program()

```
Qspi_Ip_StatusType Qspi_Ip_Program (
            uint32 instance,
            uint32 address,
            const uint8 * data,
            uint32 size )
```

Writes data in serial flash.

Parameters

instance	External flash instance number
address	Start address of area to be programmed
data	Data to be programmed in flash
size	Size of data buffer

Returns

Error or success status returned by API

6.4.6.20 Qspi_Ip_RunCommand()

```
Qspi_Ip_StatusType Qspi_Ip_RunCommand (
            uint32 instance,
            uint16 lut,
            uint32 addr )
```

Launches a simple command for the serial flash.

Parameters

instance	External flash instance number
lut	Index of command in virtual LUT
addr	Address used in the command, or base address of the target serial flash

Returns

Error or success status returned by API

6.4.6.21 Qspi_Ip_RunReadCommand()

```
Qspi_Ip_StatusType Qspi_Ip_RunReadCommand (
            uint32 instance,
            uint16 lut,
            uint32 addr,
            uint8 * dataRead,
            const uint8 * dataCmp,
            uint32 size )
```

Launches a read command for the serial flash.

This function can launch a read command in 3 modes:

- normal read (dataRead != NULL_PTR): Data is read from serial flash and placed in the buffer
- verify (dataRead == NULL_PTR, dataCmp != NULL_PTR): Data is read from serial flash and compared to the reference buffer
- blank check (dataRead == NULL_PTR, dataCmp == NULL_PTR): Data is read from serial flash and compared to 0xFF Only normal read mode can use DMA.

Parameters

instance	External flash instance number
lut	Index of LUT register
addr	Start address for read operation in serial flash
dataRead	Buffer where to store read data
dataCmp	Buffer to be compared to read data
size	Size of data buffer

Returns

Error or success status returned by API

6.4.6.22 Qspi_Ip_RunWriteCommand()

```
Qspi_Ip_StatusType Qspi_Ip_RunWriteCommand (
            uint32 instance,
            uint16 lut,
            uint32 addr,
            const uint8 * data,
            uint32 size )
```

Launches a write command for the serial flash.

Parameters

instance	External flash instance number
lut	Index of LUT register
addr	Start address for write operation in serial flash
data	Data to be programmed in flash
size	Size of data buffer

Returns

Error or success status returned by API

6.4.6.23 Qspi_Ip_AhbReadEnable()

```
Qspi_Ip_StatusType Qspi_Ip_AhbReadEnable (
            uint32 instance )
```

Sets up AHB reads to the serial flash.

Parameters

instance	External flash instance number

Returns

Error or success status returned by API

6.4.6.24 Qspi_Ip_ControllerGetStatus()

```
Qspi_Ip_StatusType Qspi_Ip_ControllerGetStatus (
            uint32 instance )
```

Check the status of the QSPI controller.

Parameters

instance	QSPI peripheral instance number

Returns

Error or success status returned by API

6.4.6.25 Qspi_Ip_ControllerInit()

```
Qspi_Ip_StatusType Qspi_Ip_ControllerInit (
             uint32 instance,
             const Qspi_Ip_ControllerConfigType * userConfigPtr )
```

Initializes the qspi driver.

This function initializes the qspi driver and prepares it for operation.

Parameters

instance	QSPI peripheral instance number
userConfigPtr	Pointer to the qspi configuration structure.

Returns

Error or success status returned by API

6.4.6.26 Qspi_Ip_ControllerDeinit()

```
Qspi_Ip_StatusType Qspi_Ip_ControllerDeinit (
             uint32 instance )
```

De-initialize the qspi driver.

This function de-initializes the qspi driver. The driver can't be used again until reinitialized. The context structure is no longer needed by the driver and can be freed after calling this function.

Parameters

instance	QSPI peripheral instance number

Returns

Error or success status returned by API

6.4.6.27 Qspi_Ip_ReadSfdp()

```
Qspi_Ip_StatusType Qspi_Ip_ReadSfdp (
            Qspi_Ip_MemoryConfigType * pConfig,
            const Qspi_Ip_MemoryConnectionType * pConnect )
```

Initializes the serial flash memory configuration from SFDP table.

This function uses the information in the SFDP table to auto-fill the memory configuration structure.

Parameters

pConfig	Pointer to the driver configuration structure.
pConnect	Pointer to the flash device connection structure.

Returns

Error or success status returned by API

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support