

User Manual

for S32K3 MEM_EXFLS Driver

Document Number: UM34MEM_EXFLS ASRR21-11 Rev0000R3.0.0 Rev. 1.0

1 Revision History	2
2 Introduction	3
2.1 Supported Derivatives	3
2.2 Overview	4
2.3 About This Manual	5
2.4 Acronyms and Definitions	6
2.5 Reference List	6
3 Driver	7
3.1 Requirements	7
3.2 Driver Design Summary	7
3.2.1 Mem configuration	7
3.2.2 Job management	8
3.2.3 Multi-instance	9
3.2.4 Burst modes	10
3.2.5 ECC handling	11
3.2.6 QuadSPI driver architecture	13
3.2.7 HyperFlash Usage	29
3.3 Hardware Resources	32
3.3.1 QuadSPI hardware instances	32
3.3.2 QuadSPI AHB regions	33
3.3.3 QuadSPI clocking	33
3.3.4 QuadSPI pin mux	35
3.3.5 QuadSPI supported read modes	37
3.4 Deviations from Requirements	38
3.5 Driver Limitations	43
3.6 Driver usage and configuration tips	43
3.7 Runtime errors	44
3.7.1 Transient Faults	44
3.7.2 Development Errors	44
3.8 Symbolic Names Disclaimer	44
4 Tresos Configuration Plug-in	45
4.1 Module Mem	49
4.2 Container AutosarExt	50
4.3 Parameter MemEnableUserModeSupport	50
4.4 Container MemGeneral	51
4.5 Parameter MemDevErrorDetect	51
4.6 Parameter MemQspiInitCallout	52
4.7 Parameter MemQspiResetCallout	52

4.8 Parameter MemQspiErrorCheckCallout	53
4.9 Parameter MemQspiEccCheckCallout	54
4.10 Parameter MemExFlsTimeoutMethod	54
4.11 Parameter MemQspiSyncReadTimeout	55
4.12 Parameter MemQspiAsyncWriteTimeout	55
4.13 Parameter MemQspiAsyncEraseTimeout	56
4.14 Parameter MemQspiSyncWriteTimeout	56
4.15 Parameter MemQspiSyncEraseTimeout	57
4.16 Parameter MemQspiDllLockTimeout	57
4.17 Parameter MemQspiCommandCompleteTimeout	58
4.18 Parameter MemQspiResetTimeout	58
4.19 Parameter MemQspiSfpEnableGlobal	59
4.20 Parameter MemQspiSfpEnableMdad	59
4.21 Parameter MemQspiSfpEnableFrad	60
4.22 Parameter MemQspiIdleTimeout	60
4.23 Parameter MemQspiFlashInitTimeout	60
4.24 Parameter MemQspiSoftwareResetDelay	61
4.25 Parameter MemQspiTxBufferResetDelay	61
4.26 Parameter MemQspiWriteEnableRetries	62
4.27 Container MemInstance	62
4.28 Parameter MemInstanceId	63
4.29 Reference MemDeviceRef	63
4.30 Container MemSectorBatch	64
4.31 Parameter MemStartAddress	64
4.32 Parameter MemNumberOfSectors	65
4.33 Parameter MemEraseSectorSize	65
4.34 Parameter MemReadPageSize	66
4.35 Parameter MemWritePageSize	66
4.36 Parameter MemSpecifiedEraseCycles	67
4.37 Container MemBurstSettings	67
4.38 Parameter MemEraseBurstSize	68
4.39 Parameter MemReadBurstSize	68
4.40 Parameter MemWriteBurstSize	69
4.41 Container MemDevice	69
4.42 Container MemCfg	70
4.43 Parameter MemName	70
4.44 Parameter MemAlignment	70
4.45 Parameter MemAHBReadEnable	71
4.46 Parameter MemUseSfdp	71
4.47 Parameter MemConnectionType	72

4.48 Reference MemCfgRef	73
4.49 Reference MemQspiInstance	73
4.50 Container MemSerialFlashCfg	73
4.51 Parameter MemCfgSize	74
4.52 Parameter MemCfgPageSize	74
4.53 Reference MemCfgReadLUT	75
4.54 Reference MemCfgWriteLUT	75
4.55 Reference MemCfgRead0xxLUT	76
4.56 Reference MemCfgRead0xxLUTAHB	76
4.57 Reference MemCtrlAutoCfgPtr	77
4.58 Container MemCfgReadIdSettings	77
4.59 Parameter MemCfgReadIdSize	78
4.60 Parameter MemQspiDeviceId	78
4.61 Reference MemCfgReadIdLUT	79
4.62 Container MemCfgEraseSettings	79
4.63 Parameter MemCfgErase1Size	79
4.64 Parameter MemCfgErase2Size	80
4.65 Parameter MemCfgErase3Size	80
4.66 Parameter MemCfgErase4Size	81
4.67 Reference MemCfgErase1LUT	81
4.68 Reference MemCfgErase2LUT	82
4.69 Reference MemCfgErase3LUT	82
4.70 Reference MemCfgErase4LUT	83
4.71 Reference MemChipEraseLUT	83
4.72 Container MemStatusConfig	83
4.73 Parameter MemRegSize	84
4.74 Parameter MemBusyOffset	84
4.75 Parameter MemBusyValue	85
4.76 Parameter MemWriteEnableOffset	85
4.77 Parameter MemBlockProtectionOffset	86
4.78 Parameter MemBlockProtectionWidth	86
4.79 Parameter MemBlockProtectionValue	86
4.80 Reference MemStatusRegInitReadLut	87
4.81 Reference MemStatusRegReadLut	87
4.82 Reference MemStatusRegWriteLut	88
4.83 Reference MemWriteEnableSRLut	88
4.84 Reference MemWriteEnableLut	89
4.85 Container MemSuspendSettings	89
4.86 Reference MemEraseSuspendLut	90
4.87 Reference MemEraseResumeLut	90

4.88 Reference MemProgramSuspendLut	90
4.89 Reference MemProgramResumeLut	91
4.90 Container MemResetSettings	91
4.91 Parameter MemResetCmdCount	92
4.92 Reference MemResetCmdLut	92
4.93 Container MemInitResetSettings	92
4.94 Parameter MemResetCmdCount	93
4.95 Reference MemResetCmdLut	93
4.96 Container MemInitConfiguration	94
4.97 Parameter opType	94
4.98 Parameter addr	95
4.99 Parameter size	95
4.100 Parameter shift	96
4.101 Parameter width	96
4.102 Parameter value	97
4.103 Reference MemCommand1Lut	97
4.104 Reference MemCommand2Lut	98
4.105 Reference MemWeLut	98
4.106 Reference MemCtrlCfgPtr	98
4.107 Container MemLUT	99
4.108 Parameter MemLUTIndex	99
4.109 Container MemInstructionOperandPair	100
4.110 Parameter MemInstrOperPairIndex	100
4.111 Parameter MemLUTInstruction	101
4.112 Parameter MemLUTPad	102
4.113 Parameter MemLUTOperand	102
4.114 Container MemHyperFlashCfg	103
4.115 Parameter MemCfgSize	103
4.116 Parameter MemCfgPageSize	104
4.117 Parameter MemOutputDriverStrength	104
4.118 Parameter MemRWDSLWOnDualError	105
4.119 Parameter MemSecureRegionUnlocked	105
4.120 Parameter MemReadLatency	106
4.121 Parameter MemParamSectorMap	107
4.122 Reference MemCtrlAutoCfgPtr	108
4.123 Container MemCfgReadIdSettings	108
4.124 Parameter MemCfgReadIdLUT	108
4.125 Parameter MemCfgReadIdWordAddr	109
4.126 Parameter MemCfgReadIdSize	109
4.127 Parameter MemQspiDeviceId	110

4.128 Container MemController	111
4.129 Parameter MemControllerName	111
4.130 Reference MemControllerCfgRef	111
4.131 Container MemControllerCfg	112
4.132 Parameter MemHwUnitReadMode	112
4.133 Parameter MemSerialFlashA1Size	113
4.134 Parameter MemSerialFlashA2Size	113
4.135 Parameter MemHwUnitSamplingModeA	114
4.136 Parameter MemIdleSignalDriveIOFA3HighLvl	114
4.137 Parameter MemIdleSignalDriveIOFA2HighLvl	115
4.138 Parameter MemHwUnitSamplingEdge	115
4.139 Parameter MemHwUnitSamplingDly	116
4.140 Parameter MemHwUnitTdh	116
4.141 Parameter MemHwUnitTcsh	117
4.142 Parameter MemHwUnitTcss	117
4.143 Parameter MemHwUnitColumnAddressWidth	118
4.144 Parameter MemHwUnitByteSwapping	119
4.145 Parameter MemHwUnitWordAddressable	119
4.146 Container MemAhbBuffer	120
4.147 Parameter MemAhbBufferInstance	120
4.148 Parameter MemAhbBufferMasterId	121
4.149 Parameter MemAhbBufferSize	121
4.150 Parameter MemAhbBufferAllMasters	122
4.151 Container MemDlICfgA	122
4.152 Parameter MemDlICfgADllMode	122
4.153 Parameter MemDlICfgADllCraFreqEn	123
4.154 Parameter MemDlICfgADllCraReferenceCounter	123
4.155 Parameter MemDlICfgADllCraResolution	124
4.156 Parameter MemDlICfgADllCraSlvFineOffset	124
4.157 Parameter MemDlICfgADllCraSlvDlyOffset	125
4.158 Parameter MemDlICfgADllCraSlvDlyCoarse	125
4.159 Parameter MemDlICfgADllTapSelect	126
4.160 Container MemSecureFlashProtection	126
4.161 Parameter MemQspiSfpMasterTimeout	127
4.162 Container MemQspiSfpMdadTG	127
4.163 Parameter Valid	128
4.164 Parameter SecureAttribute	128
4.165 Parameter MaskType	128
4.166 Parameter Mask	129
4.167 Parameter DomainID	129

4.168 Container MemQspiSfpFrad	130
4.169 Parameter Valid	130
4.170 Parameter StartAddress	131
4.171 Parameter EndAddress	131
4.172 Parameter ExclusiveAccessLock	131
4.173 Parameter ExclusiveAccessOwner	132
4.174 Parameter Md0Acp	132
4.175 Parameter Md1Acp	134
4.176 Container MemPublishedInformation	136
4.177 Parameter MemErasedValue	136
4.178 Container CommonPublishedInformation	136
4.179 Parameter ArReleaseMajorVersion	138
4.180 Parameter ArReleaseMinorVersion	138
4.181 Parameter ArReleaseRevisionVersion	139
4.182 Parameter ModuleId	139
4.183 Parameter SwMajorVersion	140
4.184 Parameter SwMinorVersion	140
4.185 Parameter SwPatchVersion	140
4.186 Parameter VendorApiInfix	141
4.187 Parameter VendorId	142
5 Module Index	143
5.1 Software Specification	143
6 Module Documentation	144
6.1 MEM_43_EXFLS Driver	144
6.1.1 Detailed Description	144
6.1.2 Data Structure Documentation	146
6.1.3 Macro Definition Documentation	147
6.1.4 Types Reference	148
6.1.5 Enum Reference	150
6.1.6 Function Reference	150
6.2 QSPI IPV Driver	155
6.2.1 Detailed Description	155
6.2.2 Data Structure Documentation	159
6.2.3 Macro Definition Documentation	168
6.2.4 Types Reference	169
6.2.5 Enum Reference	171
6.2.6 Function Reference	180



Chapter 1

Revision History

Revision	Date	Author	Description
1.0	31.03.2023	NXP RTD Team	S32K3 Real-Time Drivers AUTOSAR 4.4 & R21-11 Version 3.0.0

Chapter 2

Introduction

- [Supported Derivatives](#)
- [Overview](#)
- [About This Manual](#)
- [Acronyms and Definitions](#)
- [Reference List](#)

This User Manual describes NXP Semiconductors' AUTOSAR Mem External Flash Driver for S32K3XX.

AUTOSAR Mem External Flash Driver configuration parameters description can be found in the [Tresos Configuration Plug-in](#) section. Deviations from the specification are described in the [Deviations from Requirements](#) section.

AUTOSAR Mem External Flash Driver requirements and APIs are described in the Mem External Flash Driver Software Specification Document (version 4.7.0) [1] and in the Modules Documentation section.

2.1 Supported Derivatives

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

- s32k310_mqfp100
- s32k310_lqfp48
- s32k311_mqfp100 / MWCT2015S_mqfp100
- s32k311_lqfp48
- s32k312_mqfp100 / MWCT2016S_mqfp100
- s32k312_mqfp172 / MWCT2016S_mqfp172
- s32k314_mqfp172
- s32k314_mapbga257

- s32k322_mqfp100 / MWCT2D16S_mqfp100
- s32k322_mqfp172 / MWCT2D16S_mqfp172
- s32k324_mqfp172 / MWCT2D17S_mqfp172
- s32k324_mapbga257
- s32k341_mqfp100
- s32k341_mqfp172
- s32k342_mqfp100
- s32k342_mqfp172
- s32k344_mqfp172
- s32k344_mapbga257
- s32k394_mapbga289
- s32k396_mapbga289
- s32k358_mqfp172
- s32k358_mapbga289
- s32k328_mqfp172
- s32k328_mapbga289
- s32k338_mqfp172
- s32k338_mapbga289
- s32k348_mqfp172
- s32k348_mapbga289
- s32m274_lqfp64
- s32m276_lqfp64

All of the above microcontroller devices are collectively named as S32K3.

Note: MWCT part numbers contain NXP confidential IP for Qi Wireless Power.

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 MEM Driver	AUTOSAR Release R21-11
2	Reference Manual	S32K3xx Reference Manual, Rev.6, Draft B, 01/2023
		S32K39 and S32K37 Reference Manual, Rev. 2 Draft A, 11/2022
3	Datasheet	S32K3xx Data Sheet, Rev. 6, 11/2022
		S32K396 Data Sheet, Rev. 1.1 — 08/2022
4	Errata	S32K358_0P14E Mask Set Errata — Rev. 28, 9/2022
		S32K396_0P40E Mask Set Errata, Rev. DEC2022, 12/2022
		S32K311_0P98C Mask Set Errata, Rev. 6/March/2023, 3/2023
		S32K312: Mask Set Errata for Mask 0P09C, Rev. 25/April/2022
		S32K342: Mask Set Errata for Mask 0P97C, Rev. 10, 11/2022
		S32K3x4: Mask Set Errata for Mask 0P55A/1P55A, Rev. 14/Oct/2022

Chapter 3

Driver

- [Requirements](#)
- [Driver Design Summary](#)
- [Hardware Resources](#)
- [Deviations from Requirements](#)
- [Driver Limitations](#)
- [Driver usage and configuration tips](#)
- [Runtime errors](#)
- [Symbolic Names Disclaimer](#)

3.1 Requirements

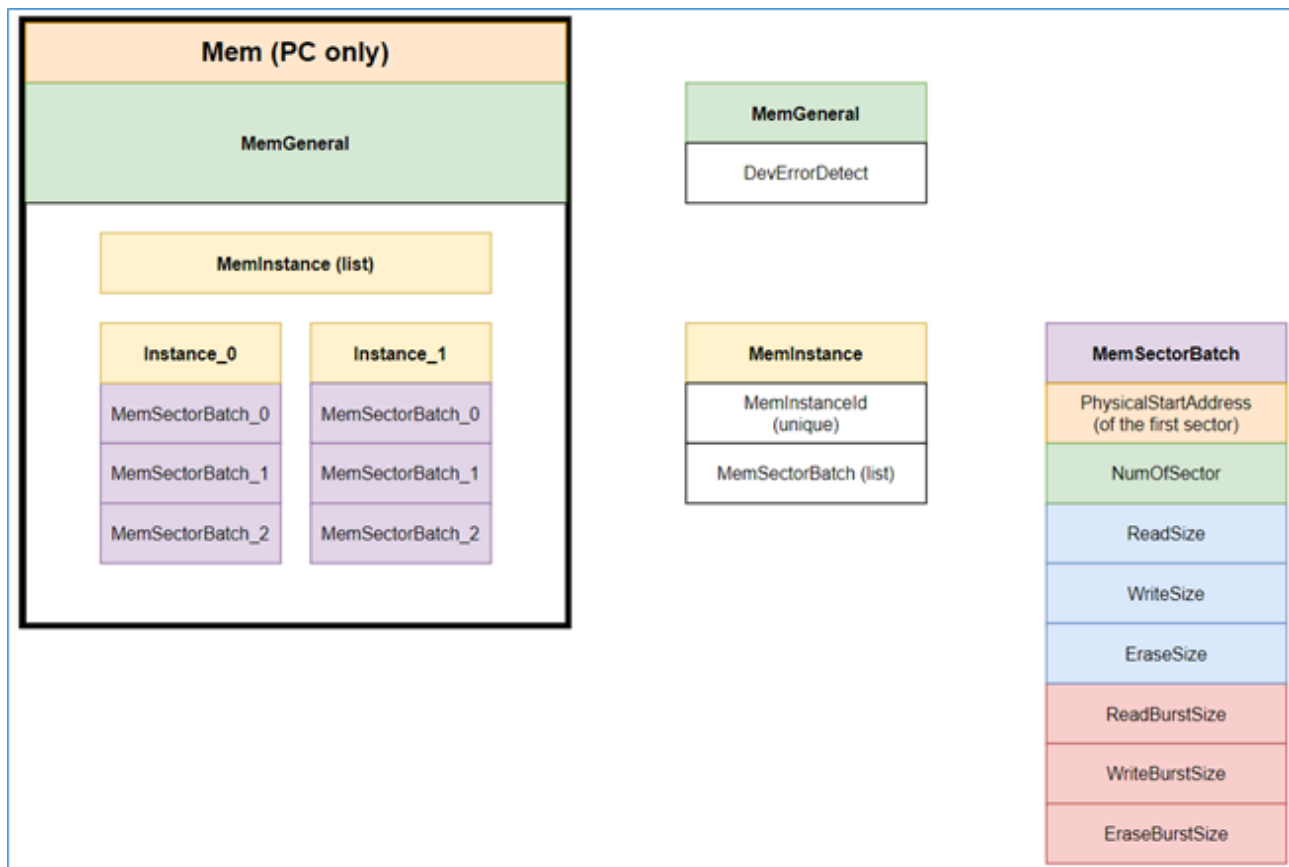
Requirements for this driver are detailed in the Autosar Driver Software Specification document (See Table Reference List).

3.2 Driver Design Summary

3.2.1 Mem configuration

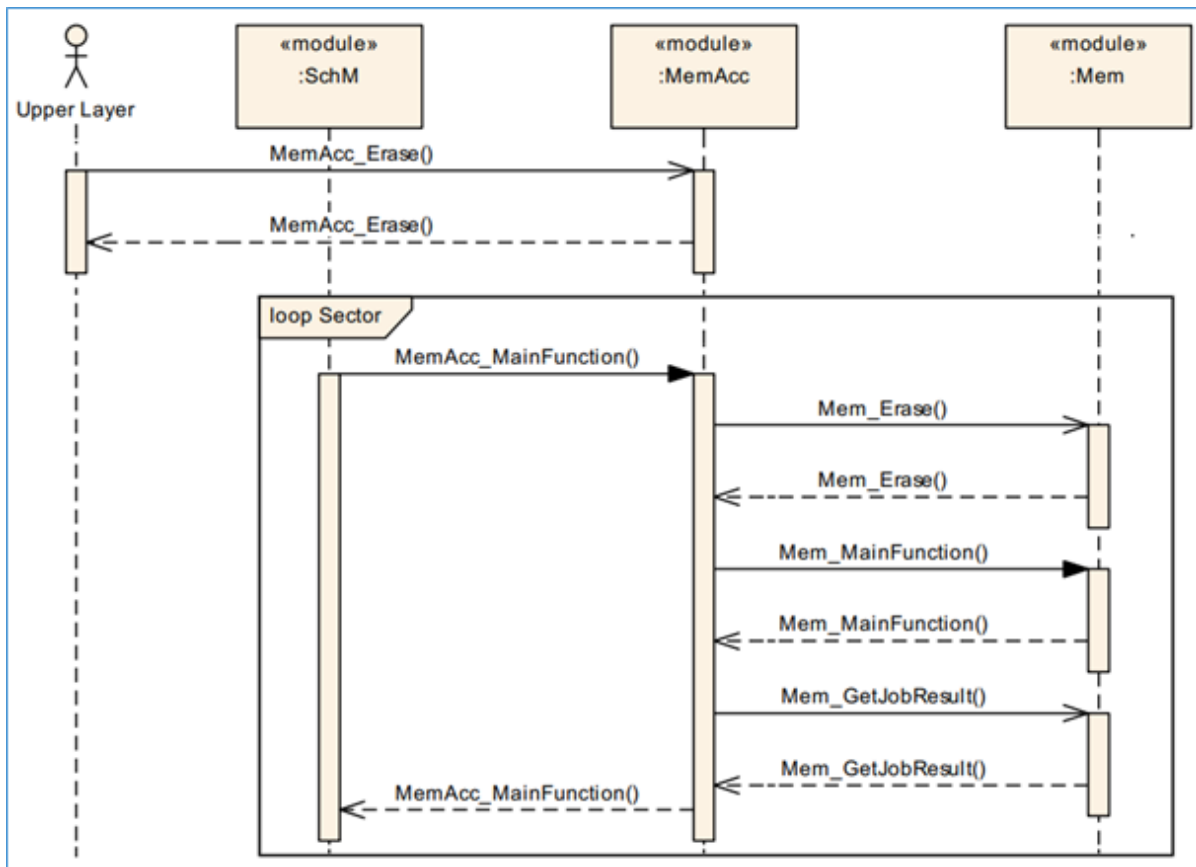
- The Mem drivers only support VARIANT-PRE-COMPILE (Pre-compile module)
- There are 2 main containers:
 - MemGeneral: pre-compile configuration parameters
 - MemInstance: includes the Mem driver instances specific configuration parameters
- MemSectorBatch: Configuration description of a programmable sector or sector batch.
 - It is recommended to group as many identical sectors as possible together

- Aggregation of sectors with uniform size
- Logical aggregation of contiguous sectors with the same size



3.2.2 Job management

- Mem drivers support basic services for reading, writing, and erasing of memory devices based on the physical segmentation
- Handle only one job at a time
- Keep track of the job processing and store result of the last job
- Do not need to measure times or do any timing supervision
- Mem drivers operates on flash pages and sectors
- Management of erasing multiple sectors or writing multiple pages is handled by MemAcc
 - Splitting of access request according to physical memory segmentation

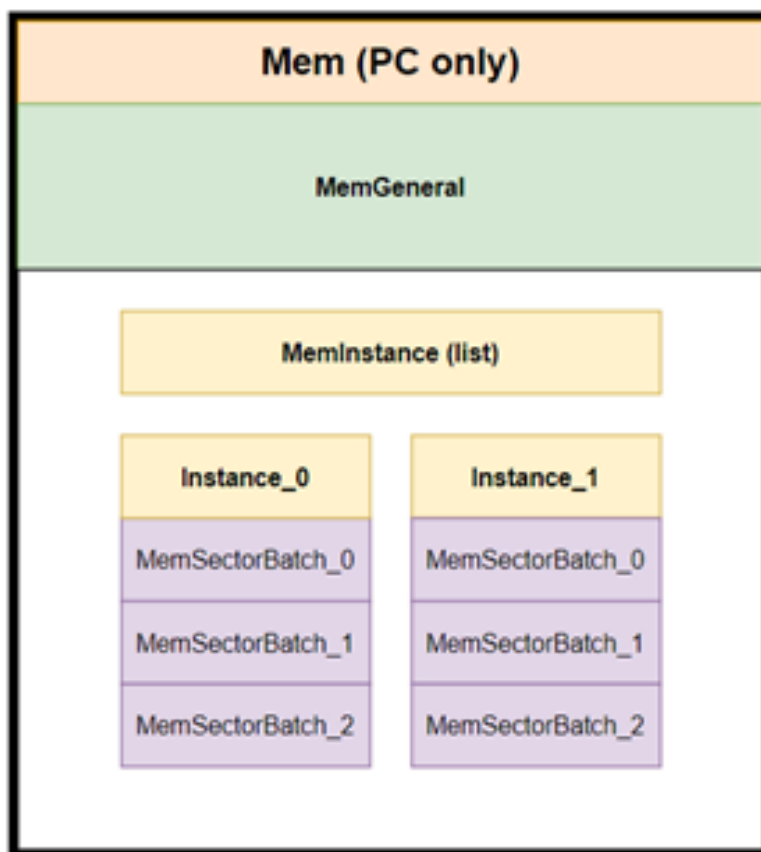


Note

If Suspend/Resume are not supported by hardware, implementation based on physical segmentation

3.2.3 Multi-instance

- Mem driver supports multiple instances of the same memory type
 - Multiple devices of the same type can be handled by one Mem driver
 - Each device uses a (different) hardware configuration
 - Distinguished by a memory instance ID

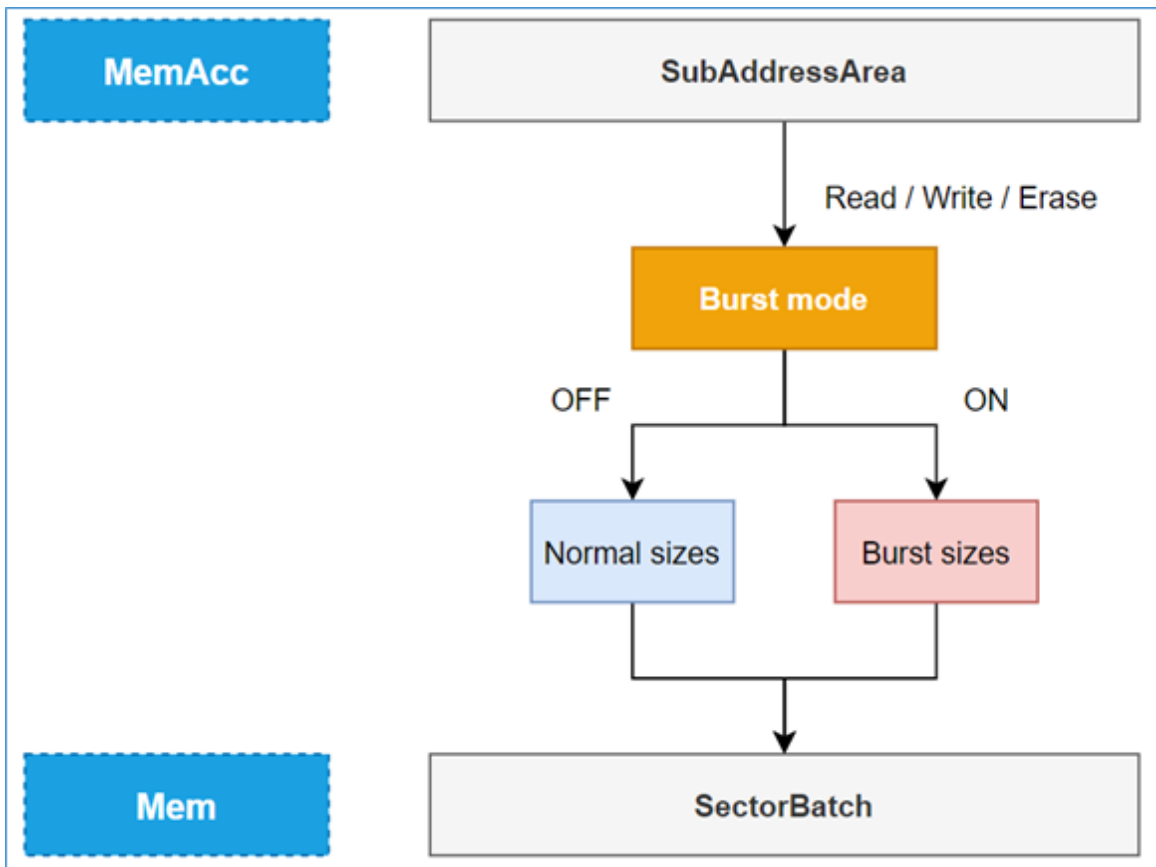


- Instance ID is passed as a parameter to select the corresponding driver instance
 - Mapping different instances to different physical memory areas
- For example: the OTA software update use case
 - Multiple memory devices of the same type are used to expand the memory resources (increase storage capacity)
 - Switching of memory address translation tables for memory swap use cases (while keep the same physical addresses)

3.2.4 Burst modes

- Some memory devices can provide burst capabilities, several pages/sectors are written/erased at once to increase performance
- Depending on hardware capabilities, Mem driver might offer three kinds of burst modes:
 - Erase multiple sectors
 - Read multiple pages
 - Write multiple pages
- If enabled, MemAcc shall align and split the job requests according to the burst size of the Mem driver

- If disabled, MemAcc uses normal sizes



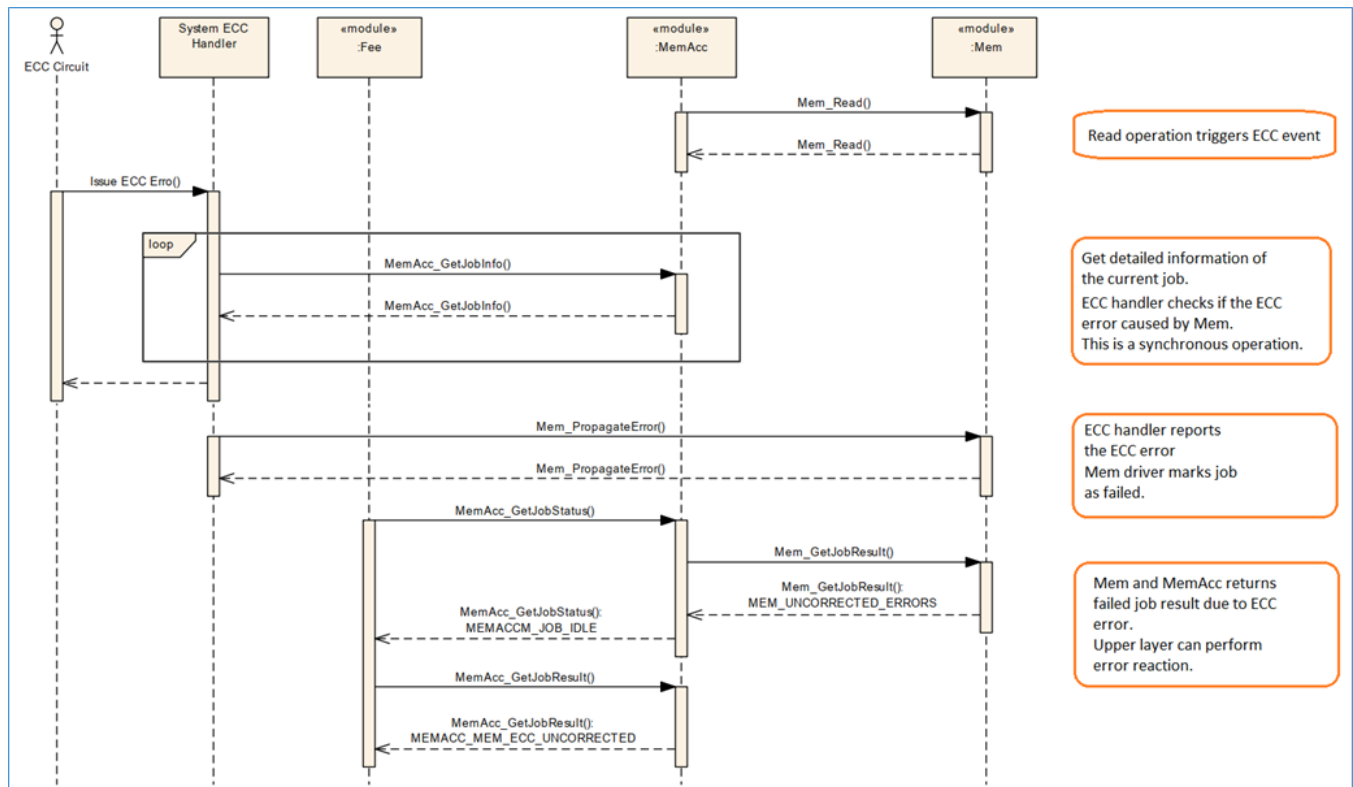
- Mem driver expects request aligned to the physical segmentation
 - Therefore, Mem needs to check the given parameters to select between burst or normal settings for the requested job

Note

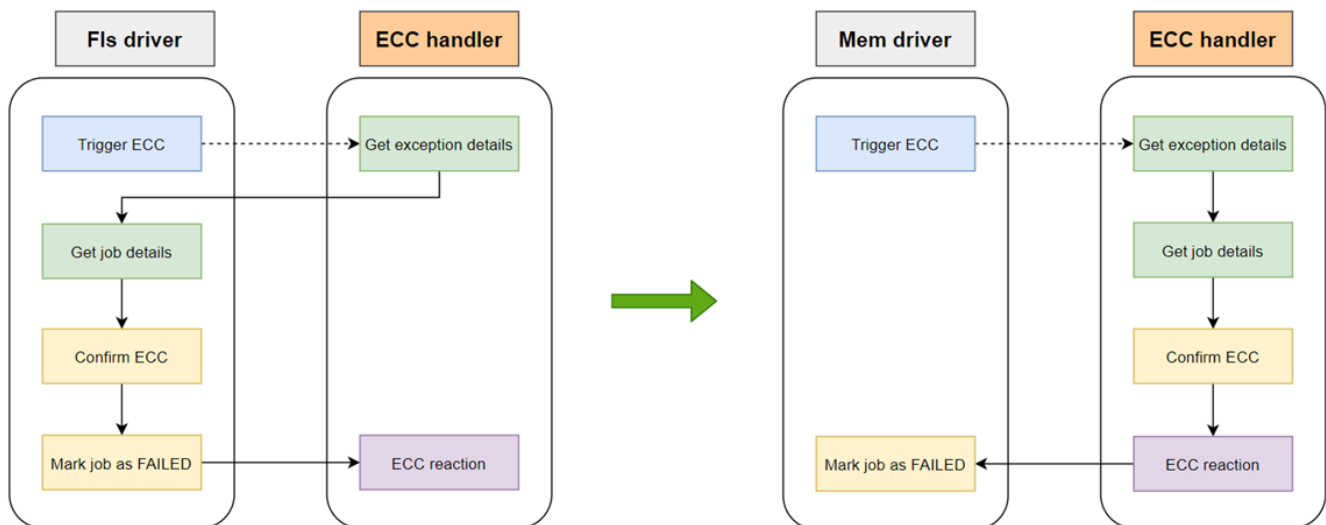
- Enabling burst mode also increases the latency when a job shall be processed with a higher priority
- Therefore, system integrators have to consider the maximum latency when configuring the burst modes

3.2.5 ECC handling

The approach of Mem driver:



- MemAcc provides MemAcc_GetJobInfo to retrieve detailed information of current job
 - Can be used by ECC error handler or any upper layer to retrieve
 - Mem_43_exfls does not handle ECC errors from IPV_QSPI, it provides the Mem_43_ExFls_↔ PropagateError to forward (spread) memory ECC errors from Mem_43_exfls to the upper layers
 - Can be called by the system ECC error handler
- The new APIs:
 - MemAcc_GetJobInfo: Get job details
 - Mem_PropagateError: Mark job as FAILED



- [QuadSPI driver architecture](#)
- [HyperFlash Usage](#)

3.2.6 QuadSPI driver architecture

This section describes the detail for a high-level overview of QuadSPI components in Mem_43_ExFls 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)

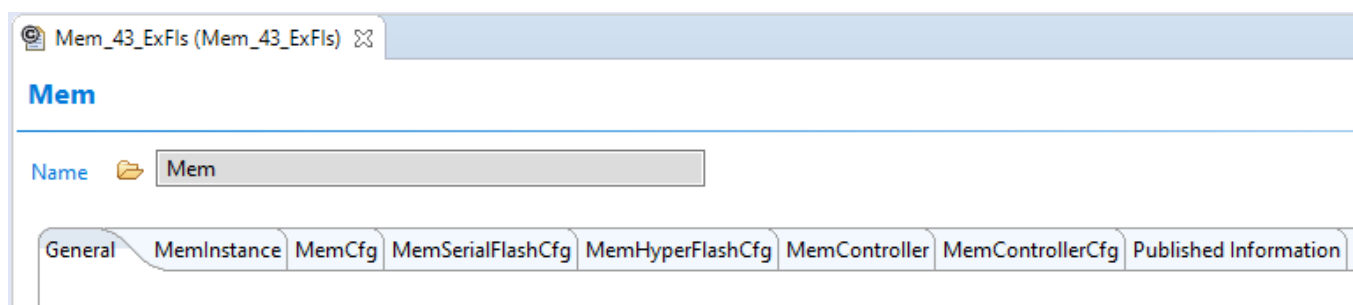
3.2.6.1 High-level overview

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

3.2.6.1.1 QuadSPI components in Mem_43_ExFls driver

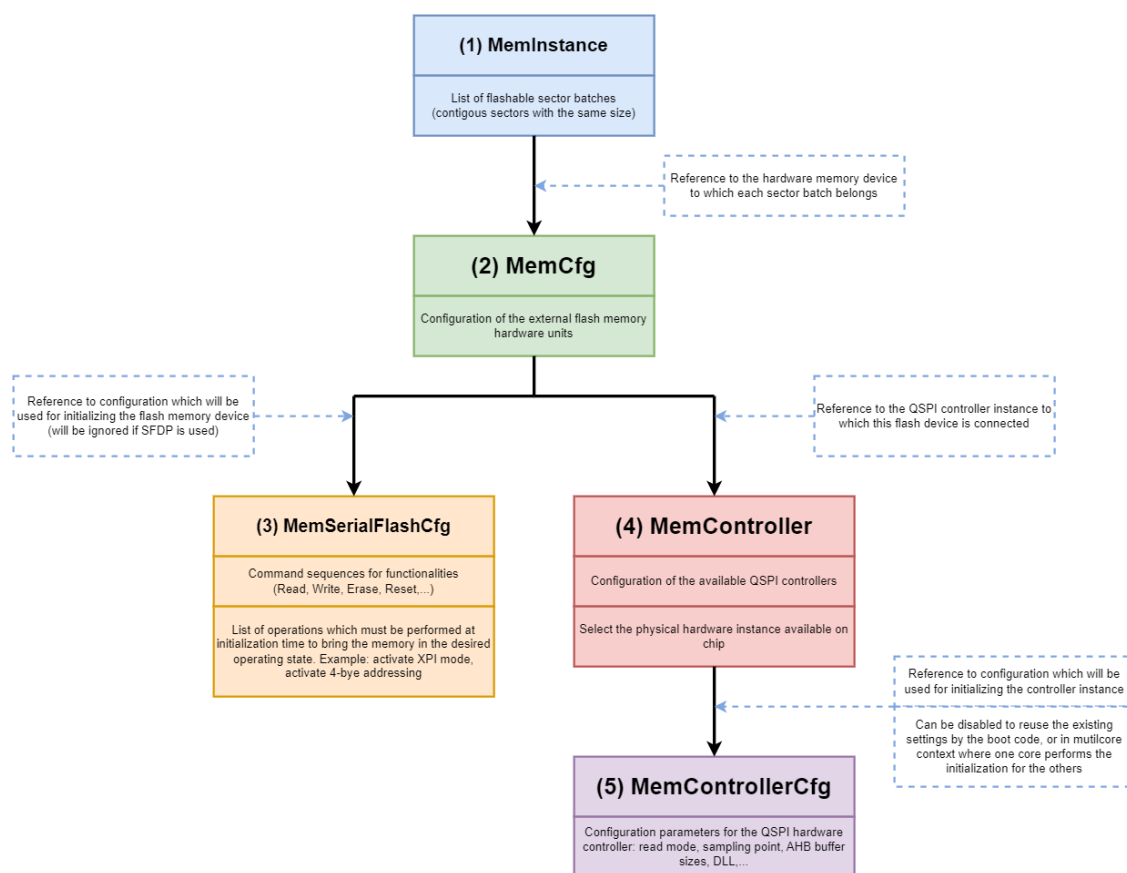
QuadSPI components user interface:



There are five layers connect together in the order below:

#	Device name
1	MemInstance
2	MemCfg
3	MemSerialFlashCfg / MemHyperFlashCfg
3	MemController
4	MemControllerCfg

Connections between components:



3.2.6.1.2 MemSerialFlashCfg

This container contains all specific settings for the memory device:

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

*Mem_43_ExFls (Mem_43_ExFls)

MemSerialFlashCfg

Name MemSerialFlashCfg_0

Memory External Flash | MemInitConfiguration | MemLUT

Flash device size (0x0 -> 0xffffffff)	0x800000
Flash device page size (0 -> 4294967295)	256
Read LUT index	/Mem_43_ExFls/Mem/MemDevice/MemSerialFlashCfg_0/Read_1S1S1S
Write LUT index	/Mem_43_ExFls/Mem/MemDevice/MemSerialFlashCfg_0/Write_1S1S1S

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:

*Mem_43_ExFls (Mem_43_ExFls)

MemInitConfiguration

Name init_0_Set_QE_bit

Initial device configuration

Operation type	QSPI_IP_OP_TYPE_RMW_REG
First LUT index	/Mem_43_ExFls/Mem/MemDevice/MemSerialFlashCfg_0/ReadStatus
Second LUT index	/Mem_43_ExFls/Mem/MemDevice/MemSerialFlashCfg_0/WriteStatus
Write Enable LUT index	/Mem_43_ExFls/Mem/MemDevice/MemSerialFlashCfg_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)	0
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

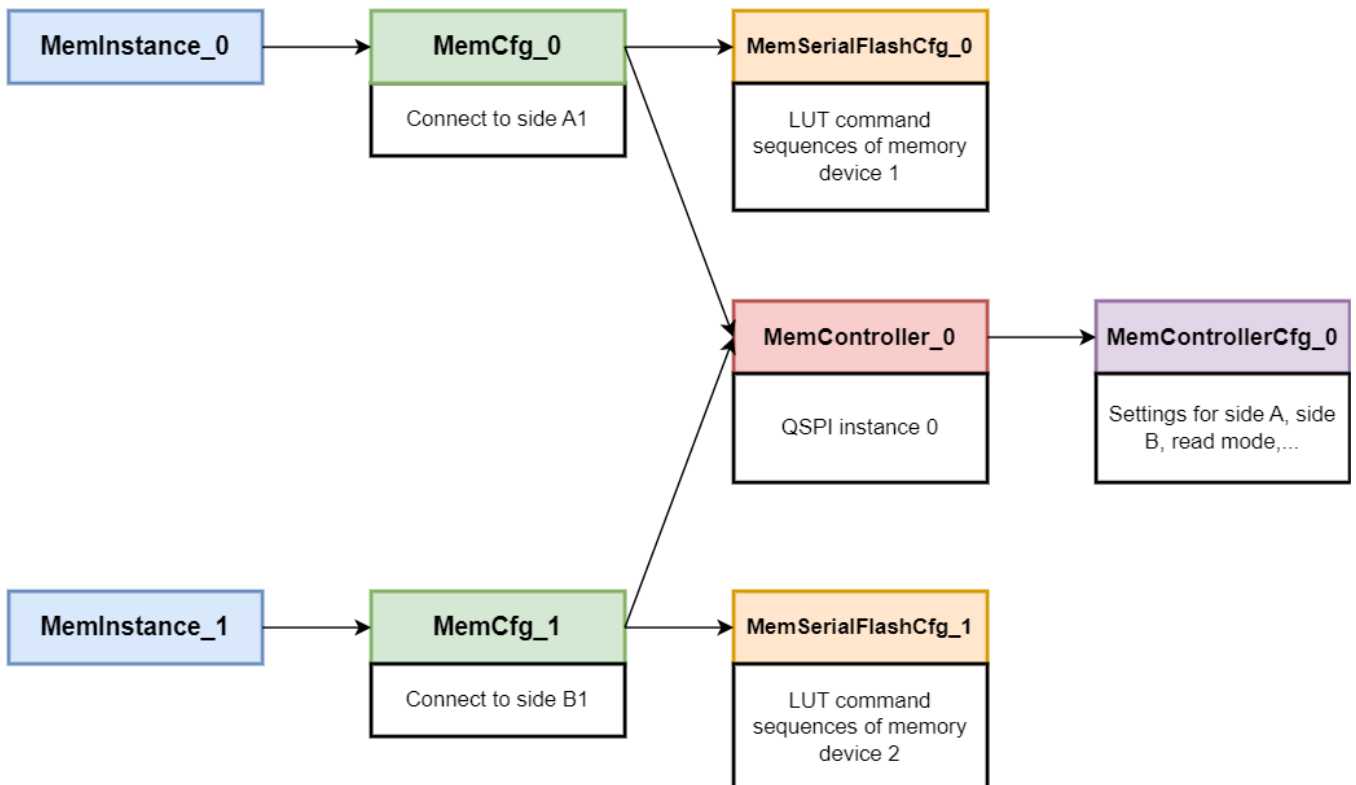
3.2.6.1.3 Examples of configuration

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

MemInstance	Mem Sector Size	Mem Number Of Sector	Mem Hardware Start Address	Mem Hardware Memory Unit
MemInstance_0	4096 (0x1000)	1	0 (0x0000)	MemCfg_0
MemInstance_1	4096 (0x1000)	1	0 (0x0000)	MemCfg_1

In this example, we need:

- 02 hardware memory unit (reference to MemSerialFlashCfg_0 and MemSerialFlashCfg_1): contain the LUT command sets for initializing each memory device (if they have different command sets)
- 01 controller configuration set (MemControllerCfg_0): contains the configuration parameters for QuadSPI hardware instance 0



As you can see:

- Any operations on the MemInstance_0 will be mapped to the hardware sector 0 (0x0000 - 0x0FFF) of the memory device 1
- Any operations on the MemInstance_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 MemSerialFlashCfg

3.2.6.2 Use cases

This sub-chapter provides various useful practical examples.

3.2.6.2.1 Software reset

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

- Software Reset (used by `Qspi_Ip_Reset()`, at any time during runtime)
- Initial Software Reset (used by `Mem_43_ExFls_Init()`, only one time)

MemSerialFlashCfg

Name: MemSerialFlashCfg_0

Memory External Flash: MemInitConfiguration | MemLUT

MemSuspendSettings

- Erase suspend LUT index
- Erase resume LUT index
- Program suspend LUT index
- Program resume LUT index

Software Reset

Name: MemResetSettings

Reset LUT index: /Mem_43_ExFls/Mem/MemDevice/MemSerialFlashCfg_0/Reset

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

Initial Software Reset

Name: MemInitResetSettings

Reset LUT index: /Mem_43_ExFls/Mem/MemDevice/MemSerialFlashCfg_0/Reset

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

Configure controller on flash init

MemLUT

Name: Reset

Mem LUT | MemInstructionOperandPair

Index	Name	A...	Memory LUT Instruction	Memory Number ...	M...
0	MemInstructionOperandPair_0	0	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_1	0x66
1	MemInstructionOperandPair_1	1	QSPI_IP_LUT_INSTR_STOP	QSPI_IP_LUT_PADS_1	0x0
2	MemInstructionOperandPair_10	10	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_4	0x99
3	MemInstructionOperandPair_11	11	QSPI_IP_LUT_INSTR_STOP	QSPI_IP_LUT_PADS_1	0x0
4	MemInstructionOperandPair_2	2	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_1	0x99
5	MemInstructionOperandPair_3	3	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_1	0x0
6	MemInstructionOperandPair_4	4	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_2	0x66
7	MemInstructionOperandPair_5	5	QSPI_IP_LUT_INSTR_STOP	QSPI_IP_LUT_PADS_1	0x0
8	MemInstructionOperandPair_6	6	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_2	0x99
9	MemInstructionOperandPair_7	7	QSPI_IP_LUT_INSTR_STOP	QSPI_IP_LUT_PADS_1	0x0
10	MemInstructionOperandPair_8	8	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_4	0x66
11	MemInstructionOperandPair_9	9	QSPI_IP_LUT_INSTR_STOP	QSPI_IP_LUT_PADS_1	0x0

Driver

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 initialization. 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: MemInitResetSettings

Reset LUT index: /Mem_43_ExFls/Mem/MemDevice/MemSerialFlashCfg_0/Reset

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

*Mem_43_ExFls (Mem_43_ExFls)

MemLUT

Name: Reset

Mem LUT: MemInstructionOperandPair

Index	Name	M...	Memory LUT Instruction	Memory Number ...	M...
0	MemInstructionOperandPair_0	0	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_1	0x66
1	MemInstructionOperandPair_1	1	QSPI_IP_LUT_INSTR_STOP	QSPI_IP_LUT_PADS_1	0x0
4	MemInstructionOperandPair_2	2	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_1	0x99
5	MemInstructionOperandPair_3	3	QSPI_IP_LUT_INSTR_STOP	QSPI_IP_LUT_PADS_1	0x0
6	MemInstructionOperandPair_4	4	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_2	0x66
7	MemInstructionOperandPair_5	5	QSPI_IP_LUT_INSTR_STOP	QSPI_IP_LUT_PADS_1	0x0
8	MemInstructionOperandPair_6	6	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_2	0x99
9	MemInstructionOperandPair_7	7	QSPI_IP_LUT_INSTR_STOP	QSPI_IP_LUT_PADS_1	0x0
10	MemInstructionOperandPair_8	8	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_4	0x66
11	MemInstructionOperandPair_9	9	QSPI_IP_LUT_INSTR_STOP	QSPI_IP_LUT_PADS_1	0x0
2	MemInstructionOperandPair_10	10	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_4	0x99
3	MemInstructionOperandPair_11	11	QSPI_IP_LUT_INSTR_STOP	QSPI_IP_LUT_PADS_1	0x0

Note

- **Software Reset** can be used to abort the on-going write/erase operations
- But, 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

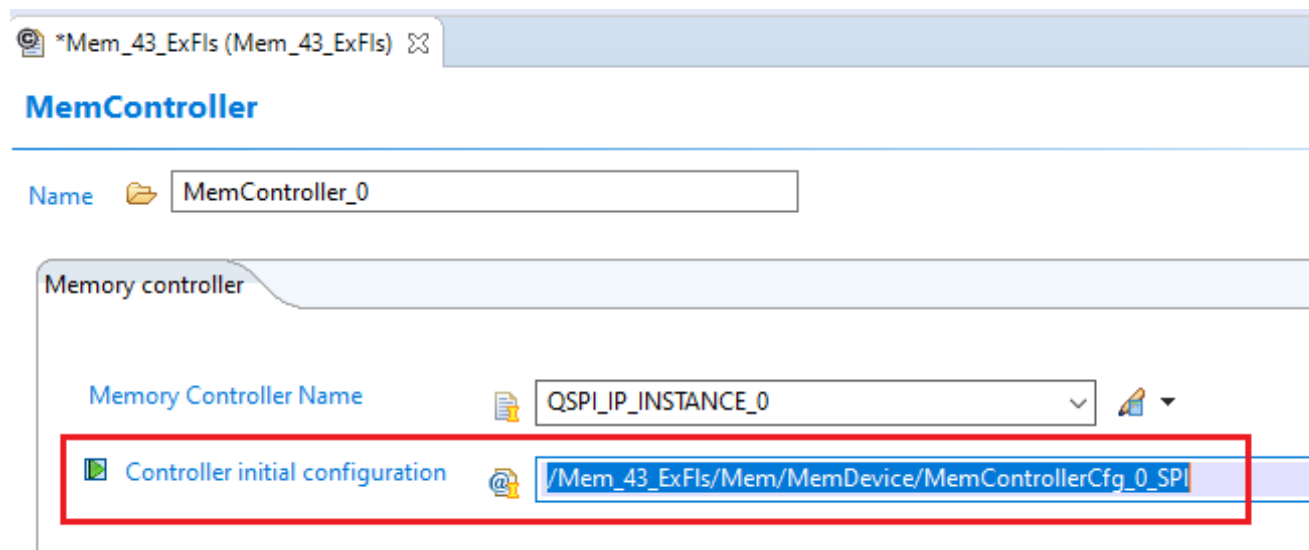
3.2.6.2.2 Controller configuration

There are several controller configuration points in the driver:

#	Controller configuration point	Location	Description
1	Controller initial configuration	MemController	Configure QuadSPI controller
2	Controller configuration	MemSerialFlashCfg (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	MemSerialFlashCfg	Re-configure QuadSPI controller when resetting the memory device E.g: switch the controller to initial configuration before re-init the memory device

3.2.6.2.2.1 Controller initial configuration

This is the first initialization point which will be done once by `Mem_43_ExFls_Init()`:




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




*Mem_43_ExFls (Mem_43_ExFls) ⌕

MemController

Name  MemController_0

Memory controller

Memory Controller Name  QSPI_IP_INSTANCE_0  ▼


 Controller initial configuration*  

3.2.6.2.2.2 Controller configuration



The second initialization point is in the list of operations of **MemSerialFlashCfg**. 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)


*Mem_43_ExFls (Mem_43_ExFls) ⌕


MemInitConfiguration


Name  init_0_Set_QE_bit


Initial device configuration


Operation type*  QSPI_IP_OP_TYPE_QSPI_CFG  ▼


☐ First LUT index  Fls/Mem/MemDevice/MemSerialFlashCfg_0/ReadStatus ▼


☐ Second LUT index  Fls/Mem/MemDevice/MemSerialFlashCfg_0/WriteStatus ▼


☐ Write Enable LUT index  Fls/Mem/MemDevice/MemSerialFlashCfg_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)  0

☐ Controller configuration  

3.2.6.2.2.3 Configure controller on flash Init

The third configuration point is at the end of **MemSerialFlashCfg**. 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)

Initial Software Reset

Name

MemInitResetSettings

Reset LUT index

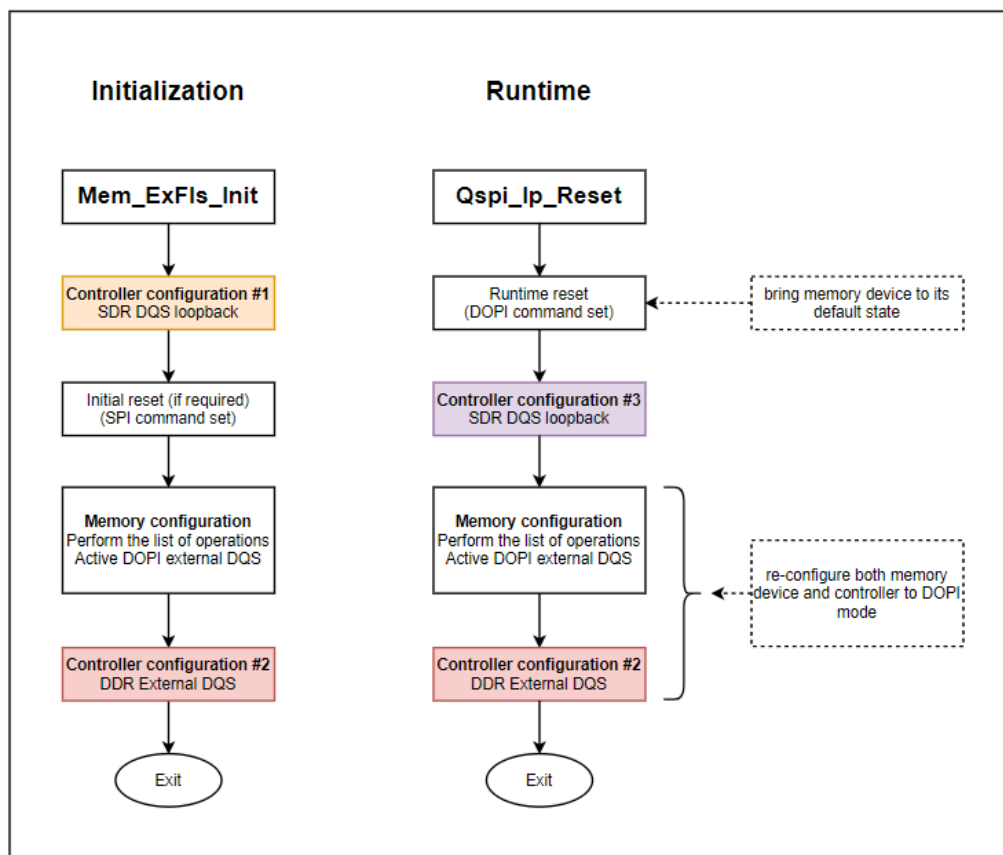
/Mem_43_ExFls/Mem/MemDevice/MemSerialFlashCfg_0/Reset

Number of reset commands (1 -> 255)

6

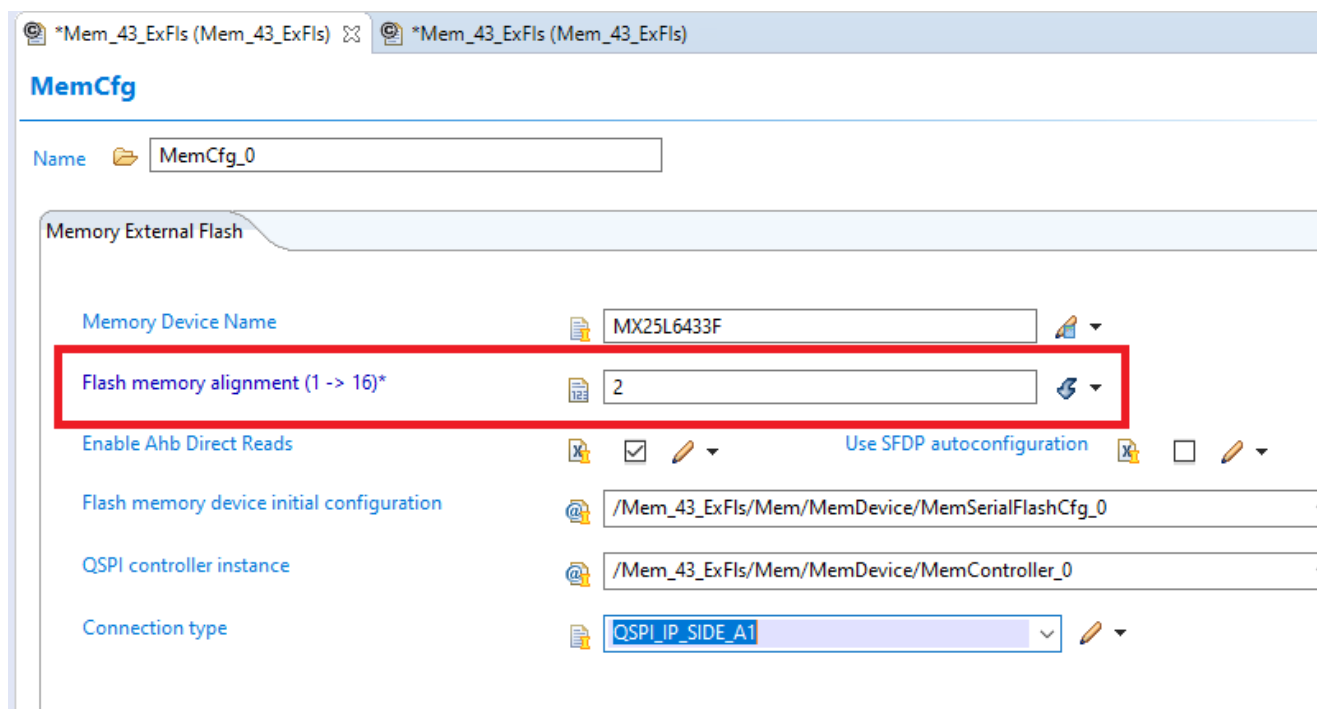
Configure controller on flash Init

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).



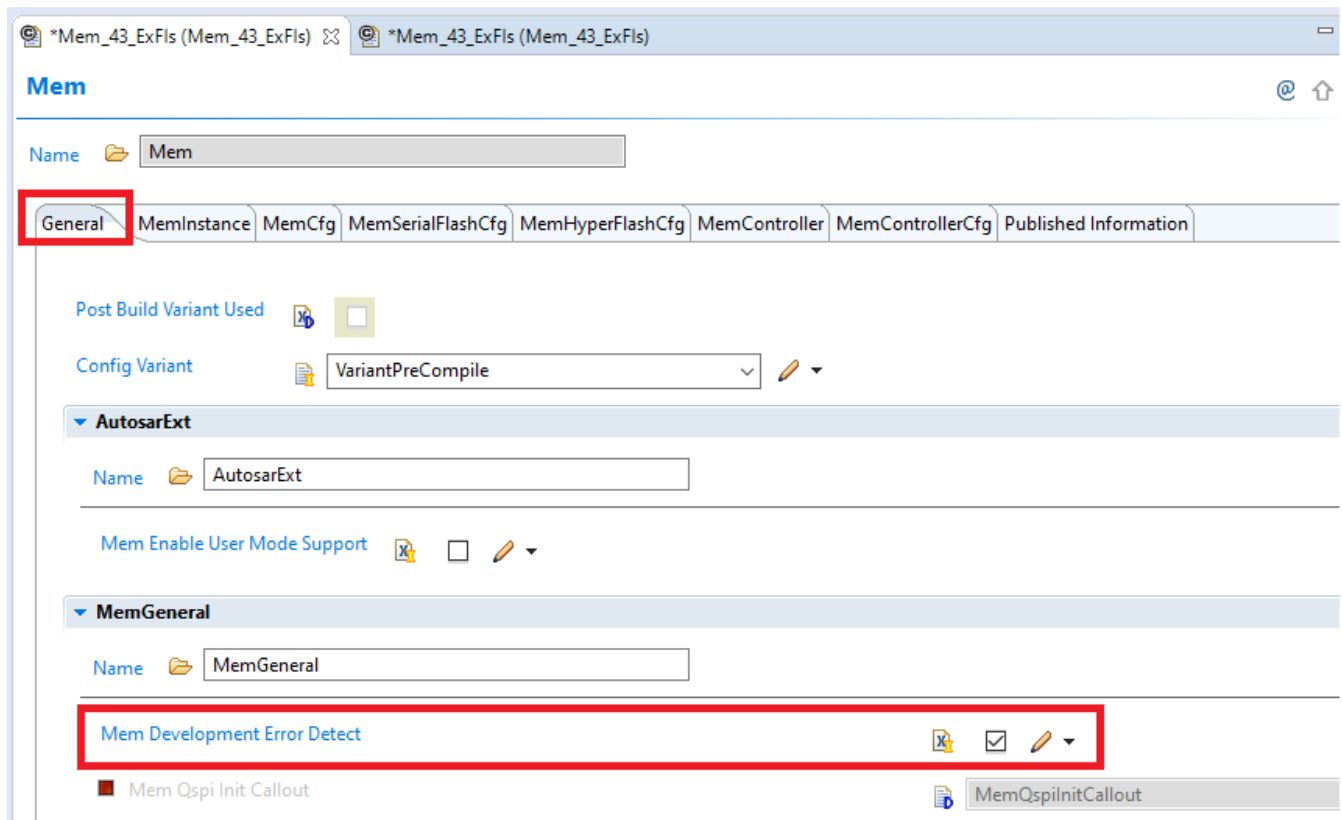
3.2.6.2.3 Read/Write from unaligned addresses

Due to the nature of DDR protocol, both the starting address must be even address and data byte number must be even. Mem_43_ExFls 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 **MemSerialFlashCfg** configuration:

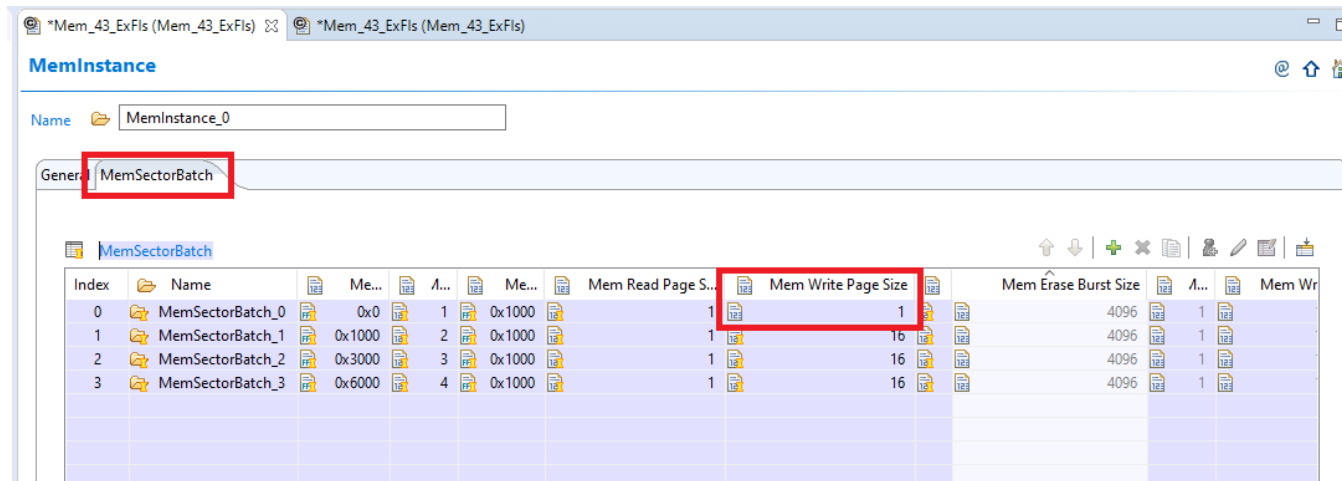


Note

- 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:

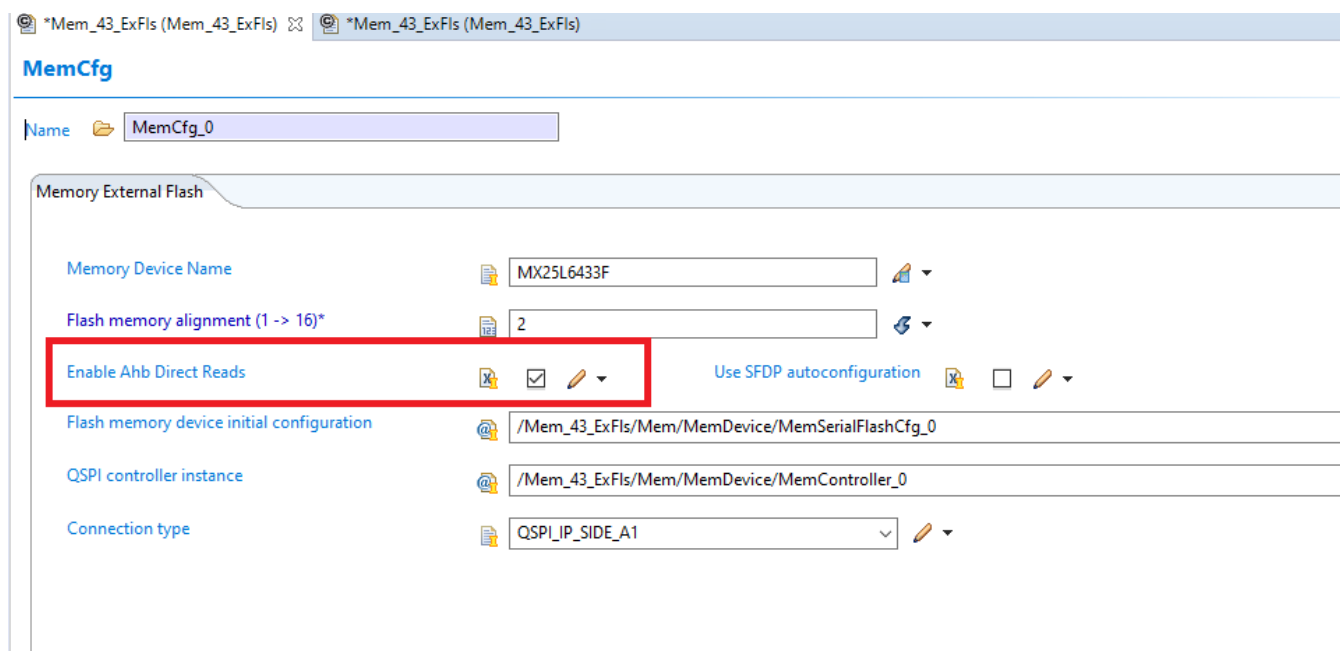


- Or configure the size of flash page boundary to 1 to meet that requirement

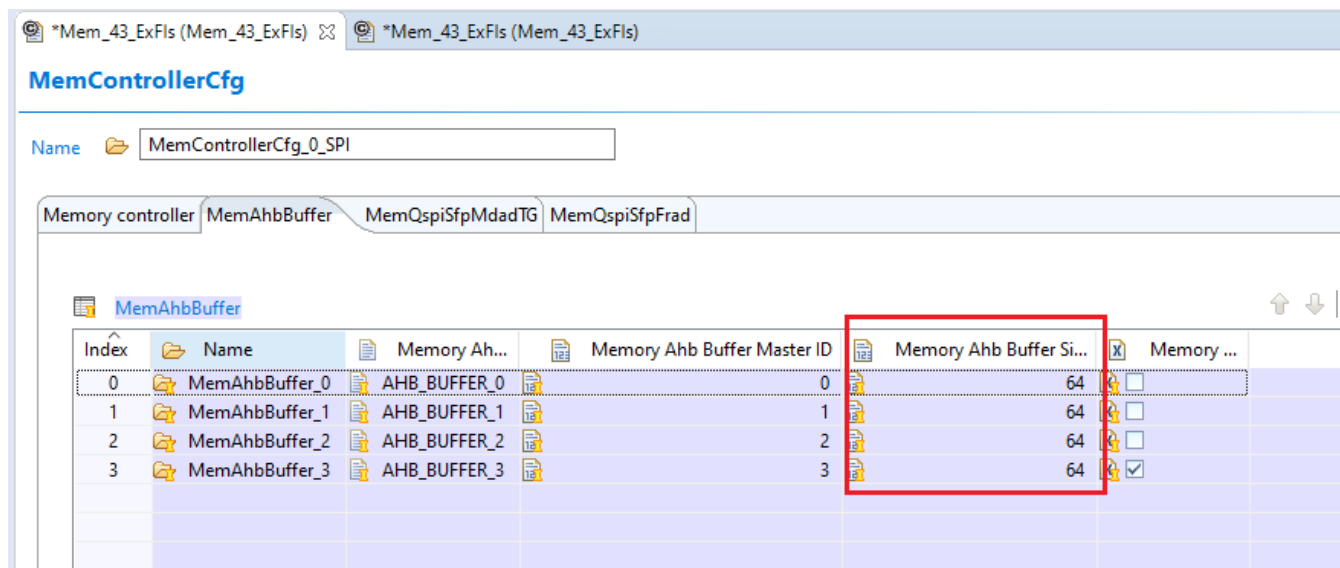


3.2.6.2.4 AHB read

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



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



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:

Driver

*Mem_43_ExFls (Mem_43_ExFls) ✕ *Mem_43_ExFls (Mem_43_ExFls)

MemSerialFlashCfg

Name

Memory External Flash MemInitConfiguration MemLUT

Flash device size (0x0 -> 0xffffffff)

Flash device page size (0 -> 4294967295)

Read LUT index

Write LUT index

*Mem_43_ExFls (Mem_43_ExFls) ✕ *Mem_43_ExFls (Mem_43_ExFls)

MemLUT

Name

Mem LUT MemInstructionOperandPair

Index	Name	Memory Instruction Op...	Memory LUT Instruction	Memory Number of LUT Pad	Memory LUT Operand
0	MemInstructionOperandPair_0	0	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_1	0x3
1	MemInstructionOperandPair_1	1	QSPI_IP_LUT_INSTR_AD...	QSPI_IP_LUT_PADS_1	0x18
2	MemInstructionOperandPair_2	2	QSPI_IP_LUT_INSTR_READ	QSPI_IP_LUT_PADS_1	0x8

3.2.6.2.5 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:

*Mem_43_ExFls (Mem_43_ExFls) ✕ *Mem_43_ExFls (Mem_43_ExFls)

MemSerialFlashCfg

Name

Memory External Flash MemInitConfiguration MemLUT

Flash device size (0x0 -> 0xffffffff)

Flash device page size (0 -> 4294967295)

Read LUT index

Write LUT index

☒ 0xx Read LUT index*

☒ 0xx Read LUT index - AHB version*

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

Name

Mem LUT MemInstructionOperandPair

Index	Name	Memory Instruction Op...	Memory LUT Instruction	Memory Number of LUT Pad	Memory LUT Operand
0	MemInstructionOperandPair_0	0	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_1	0xb
1	MemInstructionOperandPair_1	1	QSPI_IP_LUT_INSTR_AD...	QSPI_IP_LUT_PADS_1	0x18
2	MemInstructionOperandPair_2	2	QSPI_IP_LUT_INSTR_DU...	QSPI_IP_LUT_PADS_1	0x8
3	MemInstructionOperandPair_3	3	QSPI_IP_LUT_INSTR_READ	QSPI_IP_LUT_PADS_1	0x8

Name

Mem LUT MemInstructionOperandPair

Index	Name	Memory Instruction Op...	Memory LUT Instruction	Memory Number of LUT Pad	Memory LUT Operand
0	MemInstructionOperandPair_0	0	QSPI_IP_LUT_INSTR_CMD	QSPI_IP_LUT_PADS_1	0xeb
1	MemInstructionOperandPair_1	1	QSPI_IP_LUT_INSTR_AD...	QSPI_IP_LUT_PADS_4	0x18
2	MemInstructionOperandPair_2	2	QSPI_IP_LUT_INSTR_MO...	QSPI_IP_LUT_PADS_4	0x5a
3	MemInstructionOperandPair_3	3	QSPI_IP_LUT_INSTR_DU...	QSPI_IP_LUT_PADS_4	0x4
4	MemInstructionOperandPair_4	4	QSPI_IP_LUT_INSTR_READ	QSPI_IP_LUT_PADS_4	0x8
5	MemInstructionOperandPair_5	5	QSPI_IP_LUT_INSTR_IM...	QSPI_IP_LUT_PADS_4	0x1

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

Jump to instruction 1(ADDR)

Name

Mem LUT MemInstructionOperandPair

Index	Name	Memory Instruction Op...	Memory LUT Instruction	Memory Number of LUT Pad	Memory LUT Operand
0	MemInstructionOperandPair_0	0	QSPI_IP_LUT_INSTR_AD...	QSPI_IP_LUT_PADS_4	0x18
1	MemInstructionOperandPair_1	1	QSPI_IP_LUT_INSTR_MO...	QSPI_IP_LUT_PADS_4	0x5a
2	MemInstructionOperandPair_2	2	QSPI_IP_LUT_INSTR_DU...	QSPI_IP_LUT_PADS_4	0x4
3	MemInstructionOperandPair_3	3	QSPI_IP_LUT_INSTR_READ	QSPI_IP_LUT_PADS_4	0x8

No read intruction

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

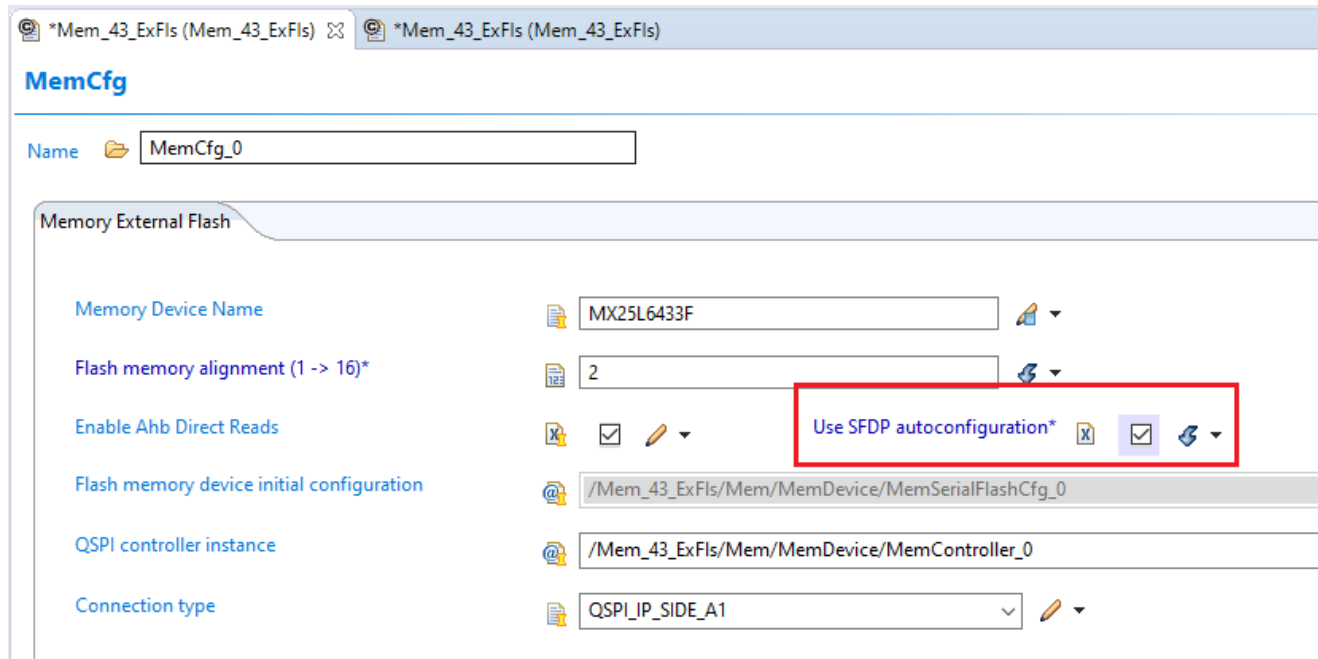
Note

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

3.2.6.2.6 SFDP feature

Mem_43_ExFls driver contains the SFDP software, user can enable this option to attempt auto-configuration using the information read from the SFDP table

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



Note

- The SFDP software only works for flash devices which support the SFDP feature
- Only standard JEDEC tables are interrogated, all vendor-specific tables will be skipped
- SFDP compliant devices must support 50 MHz operation for the Read SFDP command (instruction 5Ah). Some devices may support a wider frequency range, but user should run at 50 MHz or less and get valid results.
- The SFDP software does not support to read the tables directly in Octal-DDR mode (8D-8D-8D). It tries to issue 8D-8D-8D reset commands to force device to enter SPI mode before reading the SFDP tables. Therefore, user must enable the DDR mode in the QuadSPI controller settings for DDR instructions when working with devices boot up in Octal-DDR mode, or with Hyperflash devices which support the legacy SPI mode. (the Column Address setting is not required)

3.2.6.3 Supported memories

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

STT	Vendor	Part No	Tested on release	SFDP	Single/Quad/Octal	Densities (bit)	Voltage	DQS	Frequency Flash Specification		Frequency QSPI supported	
									SDR Freq	DDR Freq	SDR Freq	DDR Freq
1	Macronix	MX25UW51245GX D Q00	S32CC	N	Octal	512 Mb	1.8V	Y	133MHz	200MHz	133MHz	200MHz
2	Macronix	MX25UW51245GX R Q01		Y	Octal	512 Mb	1.8V	Y	133MHz	200MHz	133MHz	200MHz
3	Macronix	MX25L6433FM2 R -08G	S32K1XX S32K3XX	Y	Quad	64Mb	3V	N	133MHz	-	80MHz 120MHz	-
4	Macronix	MX25L6433FM2 I -08G	S32K3XX	Y	Quad	64Mb	3V	N	133MHz	-	120MHz	-
5	Infineon	S26HL512TC0B 00	S32K396	Y	Single Octal	512 Mb	3V 3V	N Y	166MHz -	- 200MHz	120MHz -	- 120MHz
6	ISIS	IS25LP080D-JNLE	SJA11XX	Y	Quad	8 Mb	3V	N	133MHz	66MHz	50MHz	-
7	Winbond	W25Q64JW SSI Q	S32R	Y	Quad	64 Mb	1.8V	N	133MHz	-	133MHz	66MHz
8	Macronix	MX25U6432FZNI 02		Y	Quad	64 Mb	1.8V	N	133MHZ	-	133MHZ	66MHz
9	Infineon	S26HS512TAB 00	S32ZE	Y	Single Octal	512 Mb	1.8V 1.8V	N Y	166MHz -	- 200MHz	200MHz -	- 200MHz

3.2.7 HyperFlash Usage

The Mem_43_ExFls driver supports services for initializing, reading, writing, and erasing hyperflash devices.

3.2.7.1 Memory Architecture

- HyperFlash has a sector architecture with a sector size of 256KB
- HyperFlash has an internal RAM area called Write Buffer, which is aligned to a 512-byte boundary
- Read commands are associated with pages
 - A page is a 16-word (32-byte) area of the array, on a 16-word boundary
 - Additional latency may be inserted when reading across the page boundaries
 - A Half-Page is defined as a 16-byte of memory, half the size of a page
- Multiple writes to the same half page without erase will disable the ECC functionality
 - If the two bit error detection mode is enabled, attempting to program more than once in the same half-page will result in programming operation failure status
- HyperFlash devices are word-addressable
 - The address provided for read/write operations must be the address of the word read/written multiplied by 2
 - Mem_ExFls driver supports read/write operations with unaligned addresses and sizes by using the feature **Flash Memory Alignment**
 - For example, user can even read/write 1-byte of data from/to the hyperflash memory

*Mem_43_ExFls (Mem_43_ExFls) *Mem_43_ExFls (Mem_43_ExFls)

MemCfg

Name MemCfg_0_HyperFlash

Memory External Flash

Memory Device Name HyperFlash

Flash memory alignment (1 -> 16) 2

3.2.7.2 How to configuration in EB Treos

- HyperFlash Configuration

MemHyperFlashCfg

Name MemHyperFlashCfg_0

MemHyperFlashCfg

Flash device size (0x0 -> 0xffffffff) 0x800000

Flash device page size (0 -> 4294967295) 256

Output driver strength QSPI_IP_HF_DRV_STRENGTH_000

RWDS Low On Dual Error ☐ Secure Region Unlocked ☒

Read Latency QSPI_IP_HF_READ_LATENCY_5_CLOCKS

Parameter Sector Mapping IF_UNIFORM_SECTORS_READ_PASSWORD_LOW

▼ Read Device/Manufacturer ID

Name MemCfgReadIdSettings

Read Id LUT index QSPI_IP_HF_LUT_READ

Read Id word address (0x0 -> 0xffffffff) 0x800

Read Id size (0 -> 10) 10

Mem Qspi Device Id 0x90:00:0F:00:1A:00:7B:00:34:00

Configure controller on flash Init

- Controller Configuration for HyperFlash

MemControllerCfg

Name

Memory controller | MemAhbBuffer | MemQspiSfpMdadTG | MemQspiSfpFrad

Memory Hw Unit Read Mode	<input type="text" value="QSPI_IP_DATA_RATE_DDR"/>
Memory Serial Flash A1 Size (0x0 -> 0xffffffff)	<input type="text" value="0x4000000"/>
Memory Serial Flash A2 Size (0x0 -> 0xffffffff)	<input type="text" value="0x0"/>
Memory Hw Unit Sampling Mode A	<input type="text" value="QSPI_IP_READ_MODE_EXTERNAL_DQS"/>
Idle Signal Drive IOFA3 High Level	<input checked="" type="checkbox"/> Idle Signal Drive IOFA2 High Level
Memory Hw Unit Sampling Edge	<input type="text" value="QSPI_IP_SAMPLE_PHASE_NON_INVERTED"/>
Memory Hw Unit Sampling Delay	<input type="text" value="QSPI_IP_SAMPLE_DELAY_SAME_DQS"/>
Memory Hw Unit TDH	<input type="text" value="QSPI_IP_FLASH_DATA_ALIGN_2X_REFCLK"/>
Memory Hw Unit TCSH (0 -> 15)	<input type="text" value="5"/>
Memory Hw Unit TCSS (0 -> 15)	<input type="text" value="2"/>
Memory Hw Unit Column Address Width (0 -> 15)	<input type="text" value="3"/>
Memory Hw Unit Byte Swapping	<input type="checkbox"/> <input checked="" type="checkbox"/> Memory Hw Unit Word Addressable

- MemCfg Configuration for HyperFlash

MemCfg

Name

Memory External Flash

Memory Device Name	<input type="text" value="HyperFlash"/>
Flash memory alignment (1 -> 16)	<input type="text" value="2"/>
Enable Ahb Direct Reads	<input checked="" type="checkbox"/> Use SFDP autoconfiguration
Flash memory device initial configuration	<input type="text" value="/Mem_43_ExFls/Mem/MemDevice/MemHyperFlashCfg_0"/>
QSPI controller instance	<input type="text" value="/Mem_43_ExFls/Mem/MemDevice/MemController_0"/>
Connection type	<input type="text" value="QSPI_IP_SIDE_A1"/>

- The following QuadSPI controller settings must be used when using this driver, or else the driver may not be able to communicate with the device:

- `.dataRate = QSPI_IP_DATA_RATE_DDR;`
- `.columnAddr = 3U;`
- `.wordAddressable = TRUE;`
- `.readModeA = QSPI_IP_READ_MODE_EXTERNAL_DQS;`
- `.sampleDelay = QSPI_IP_SAMPLE_DELAY_SAME_DQS;`
- `.samplePhase = QSPI_IP_SAMPLE_PHASE_NON_INVERTED;`
- `.dataAlign = QSPI_IP_FLASH_DATA_ALIGN_2X_REFCLK;`

3.2.7.3 Initialization

The HyperFlash device will be initialized by calling [Mem_43_ExFls_Init\(\)](#). It launches QuadSPI commands to change device settings if necessary, and makes sure that the HyperFlash device is properly configured to work. After the call of this function, the driver is available for read/write/erase operations on the HyperFlash device.

Note

- The memory device must be in the hyperbus mode (8-8-8 DDR)
- `Mem_43_ExFls` driver will not work if the device is in another mode, for example, the legacy SPI mode (1-1-1 SDR)

3.3 Hardware Resources

3.3.1 QuadSPI hardware instances

Note

QuadSPI is not available on S32K312

- For S32K341 / S32K322 / S32K342 / S32K314 / S32K324 / S32K344
 - The Quad Serial Peripheral Interface (QuadSPI) block acts as an interface to single serial flash devices, with up to four bidirectional data lines. The QuadSPI supports 4-pin Quad A interface only in SDR mode
- For S32K396 / S32KK358
 - The QuadSPI supports 8-pin Quad A interface in both SDR and DDR mode
 - Support for Octal flash and Hyperflash memories

3.3.2 QuadSPI AHB regions

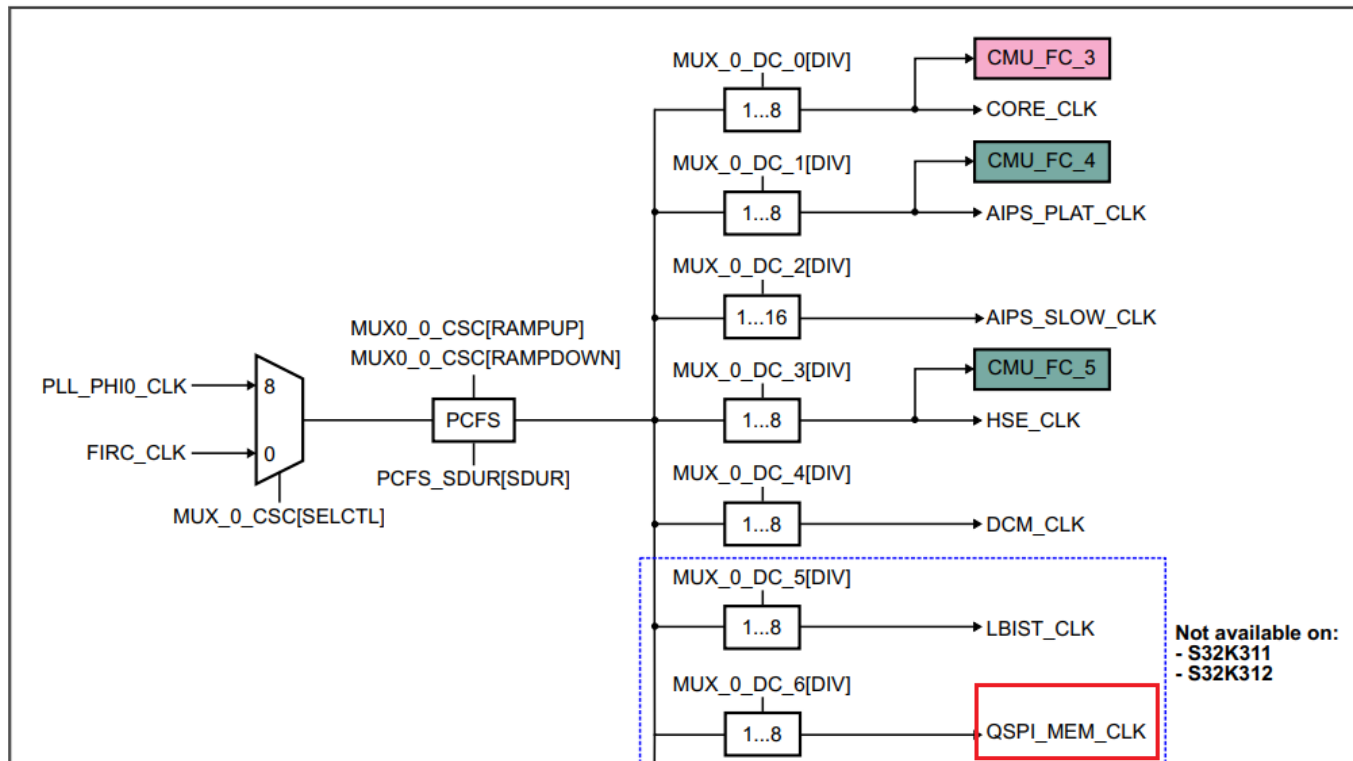
You should define valid MPU settings for the QuadSPI AHB regions accessed in the application so that CPU can access that region accordingly. If you do not define the MPU of a used region, it can cause a Translation Fault Exception.

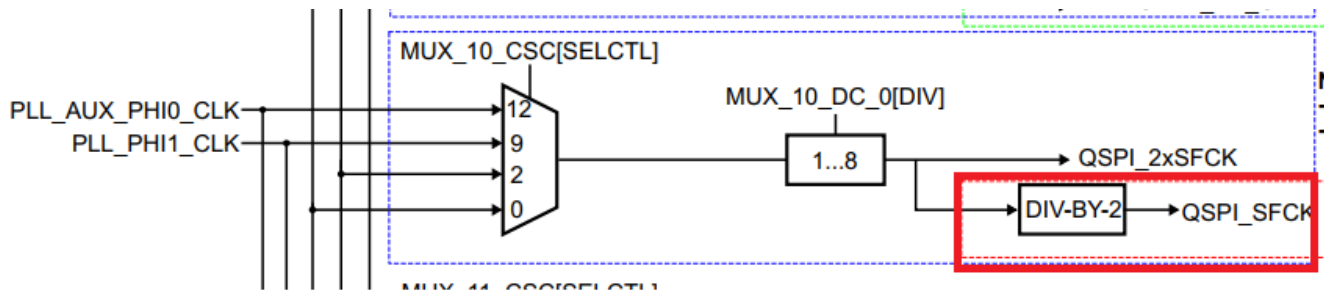
Slave Addressed by Each of the Masters				
Start	End	System (SMU, CE, DMA)	RTU0 Master	RTU1 Master
00000000	07FFFFFF	QuadSPI_0	QuadSPI_0	QuadSPI_0
08000000	0FFFFFFF			
10000000	17FFFFFF	QuadSPI_1	QuadSPI_1	QuadSPI_1

3.3.3 QuadSPI clocking

The clock signals used by QuadSPI module : QSPI_MEM_CLK, QSPI_SFCK

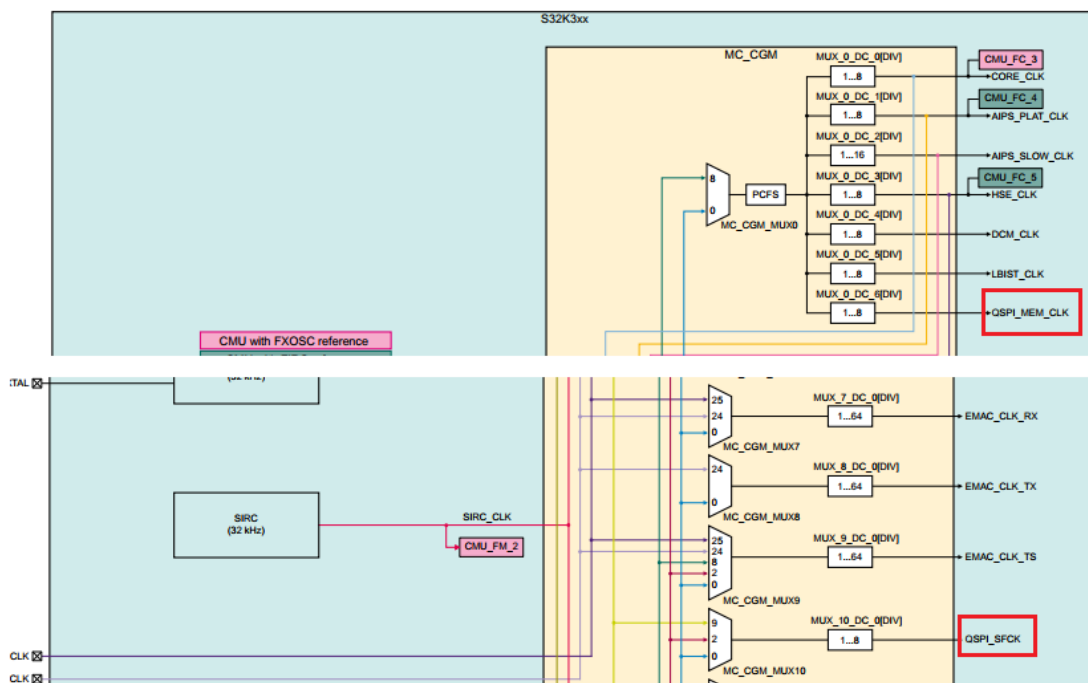
24.3.2 MC_CGM mux 0 clocks



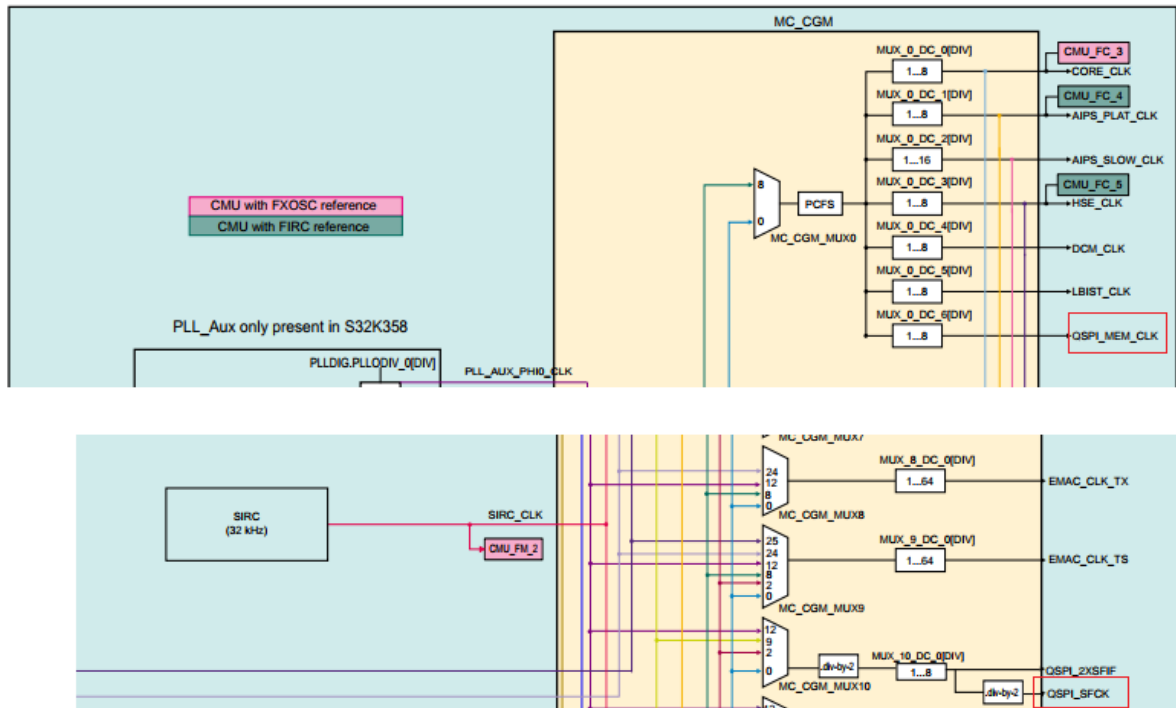


- S32K388 / S32K344 / S32K3424 / S32K314 / S32K322 / S32K341 / S32K342 : QSPI_MEM_CLK from MUX_0 and QSPI_SFCK from MUX_10 of MC_CGM

S32K344, S32K324, S32K314, S32K322, S32K341 and S32K342 clock system diagram



- S32K358 / S32K396 : QSPI_MEM_CLK from MUX_0 and QSPI_SFCK, QSPI_2XSFF from MUX_10 of MC_CGM



Note

- For the details configurations, please refer to the examples included in the Mem_ExFls plugin
- For more information about frequency range minimum and maximums, please refer to the chapter "Clock frequency ranges" in the chip data sheet

3.3.4 QuadSPI pin mux

This section provides the external signal information for the QuadSPI module.

-S32K3XX

Port	CR	SSS	Function	Description	Direction
PTC3	SIUL_MSCR67	0000_0110	QuadSPI_PCSFA	QuadSPI Chip select for serial flash device A	O
PTD10	SIUL_MSCR106	0000_0111	QuadSPI_SCKFA	QuadSPI Serial Clock for serial flash device A (fast)	O
PTD10	SIUL_IMCR821	0000_0001		QuadSPI Serial Clock for serial flash device A (fast)	I
PTD11	SIUL_MSCR107	0000_0111	QuadSPI_IOFA0	QuadSPI Serial data for serial flash device A (fast)	O
PTD11	SIUL_IMCR817	0000_0001		QuadSPI Serial data for serial flash device A (fast)	I
PTD7	SIUL_MSCR103	0000_0111	QuadSPI_IOFA1	QuadSPI Serial data for serial flash device A (fast)	O
PTD7	SIUL_IMCR818	0000_0001		QuadSPI Serial data for serial flash device A (fast)	I
PTD12	SIUL_MSCR108	0000_0111	QuadSPI_IOFA2	QuadSPI Serial data for serial flash device A (fast)	O
PTD12	SIUL_IMCR819	0000_0001		QuadSPI Serial data for serial flash device A (fast)	I
PTC2	SIUL_MSCR66	0000_0111	QuadSPI_IOFA3	QuadSPI Serial data for serial flash device A (fast)	O
PTC2	SIUL_IMCR820	0000_0001		QuadSPI Serial data for serial flash device A (fast)	I

-S32K396

Port	CR	SSS	Function	Description	Direction
PTC3	SIUL_MSCR67	0000_0110	QuadSPI_PCSFA	QuadSPI Chip select for serial flash device A	O
PTD10	SIUL_MSCR106	0000_0111	QuadSPI_SCKFA	QuadSPI Serial Clock for serial flash device A (fast)	O
PTD10	SIUL_IMCR821	0000_0001		QuadSPI Serial Clock for serial flash device A (fast)	I
PTC1	SIUL_MSCR65	0000_0011	QuadSPI_DQSFA	QuadSPI Data Strobe signal Flash A	O
PTC1	SIUL_IMCR935	0000_0001		QuadSPI Data Strobe signal Flash A	I
PTD11	SIUL_MSCR107	0000_0111	QuadSPI_IOFA0	QuadSPI Serial data for serial flash device A (fast)	O
PTD11	SIUL_IMCR817	0000_0001		QuadSPI Serial data for serial flash device A (fast)	I
PTD7	SIUL_MSCR103	0000_0111	QuadSPI_IOFA1	QuadSPI Serial data for serial flash device A (fast)	O
PTD7	SIUL_IMCR818	0000_0001		QuadSPI Serial data for serial flash device A (fast)	I
PTD12	SIUL_MSCR108	0000_0111	QuadSPI_IOFA2	QuadSPI Serial data for serial flash device A (fast)	O
PTD12	SIUL_IMCR819	0000_0001		QuadSPI Serial data for serial flash device A (fast)	I
PTC2	SIUL_MSCR66	0000_0111	QuadSPI_IOFA3	QuadSPI Serial data for serial flash device A (fast)	O
PTC2	SIUL_IMCR820	0000_0001		QuadSPI Serial data for serial flash device A (fast)	I
PTC0	SIUL_MSCR64	0000_0100	QuadSPI_IOFA4	QuadSPI Serial data for serial flash device A (fast)	O
PTC0	SIUL_IMCR931	0000_0001		QuadSPI Serial data for serial flash device A (fast)	I
PTD9	SIUL_MSCR105	0000_1000	QuadSPI_IOFA5	QuadSPI Serial data for serial flash device A (fast)	O
PTD9	SIUL_IMCR932	0000_0001		QuadSPI Serial data for serial flash device A (fast)	I
PTD8	SIUL_MSCR104	0000_1000	QuadSPI_IOFA6	QuadSPI Serial data for serial flash device A (fast)	O
PTD8	SIUL_IMCR933	0000_0001		QuadSPI Serial data for serial flash device A (fast)	I
PTC17	SIUL_MSCR81	0000_1000	QuadSPI_IOFA7	QuadSPI Serial data for serial flash device A (fast)	O
PTC17	SIUL_IMCR934	0000_0001		QuadSPI Serial data for serial flash device A (fast)	I
PTB2	SIUL_IMCR949	0000_0101	QuadSPI_INTA	QuadSPI Interrupt	I
PTB20	SIUL_IMCR949	0000_0001		QuadSPI Interrupt	I
PTB26	SIUL_IMCR949	0000_1000		QuadSPI Interrupt	I
PTB27	SIUL_IMCR949	0000_0010		QuadSPI Interrupt	I
PTC13	SIUL_IMCR949	0000_0011		QuadSPI Interrupt	I
PTC20	SIUL_IMCR949	0000_0110		QuadSPI Interrupt	I
PTC23	SIUL_IMCR949	0000_0111		QuadSPI Interrupt	I
PTD14	SIUL_IMCR949	0000_1001		QuadSPI Interrupt	I
PTE12	SIUL_IMCR949	0000_0100		QuadSPI Interrupt	I
				QuadSPI Interrupt	I

The following table lists the external signals belonging to the module in conjunction with the different modes of operation.

Signal name	Function	Direction	Description
PCSFA1	Peripheral Chip Select Flash Memory A1	O	This signal is the chip select for the serial flash memory device A1 that represents the first of the two flash memory devices that share IOFA.
PCSFA2	Peripheral Chip Select Flash Memory A2	O	This signal is the chip select for the serial flash memory device A2 that represents the the second of the two flash memory devices that share IOFA.
SCKFA	Serial Clock Flash Memory A	O	This signal is the serial clock output to the serial flash memory device A.
IOFA[7:0]	Serial I/O Flash memory A	I/O	These signals are the data I/O lines to/from the serial flash memory device A. See Driving external signals for details about the signal drive and timing behavior. Note that the signal pins of the serial flash memory device may change their function according to the SFM Command executed, leaving them as control inputs when single and dual instructions are executed. The module supports driving these inputs to dedicated values. In single I/O mode, QuadSPI drives data on IOFA[0] and expects data on IOFA[1].
DQSFA	Data Strobe signal Flash Memory A	I/O	This is the data strobe signal for port A. Some flash memory vendors provide the Data Read Strobe (DQS) signal to which the read data is aligned in DDR mode. It is also provided as an output signal during write data phase.
INTA	ECC error signal for Flash Memory A	I	Flash Memory A drives this signal to active low value in case of an ECC error.
INTB	ECC error signal for Flash Memory B	I	Flash Memory B drives this signal to active low value in case of an ECC error.

3.3.5 QuadSPI supported read modes

For more details about supported read mode and their configurations, please refer to the chapter "Peripheral specifications" in the chip data sheet

[-]	12	Peripheral Specifications
[+]	12.1	Clock and PLL Interfaces
[+]	12.2	Timer Modules
[+]	12.3	Communication Modules
[-]	12.4	Memories and Memory Interfaces
[-]	12.4.1	QuadSPI
	12.4.1.1	QuadSPI Quad 1.8V DDR 80MHz
	12.4.1.2	QuadSPI Quad 1.8V SDR 133MHz
	12.4.1.3	QuadSPI Octal 1.8V SDR 133MHz
	12.4.1.4	QuadSPI Octal 1.8V DDR 166MHz
	12.4.1.5	QuadSPI Octal 1.8V DDR 200MHz
	12.4.1.6	QuadSPI Quad 3.3V DDR 80MHz
	12.4.1.7	QuadSPI Octal 3.3V DDR 100MHz
	12.4.1.8	QuadSPI Quad 3.3V SDR 133MHz

3.4 Deviations from Requirements

The driver deviates from the AUTOSAR MEM_43_EXFLS Driver software specification in some places.

Term	Definition
N/A	Not available
N/T	Not testable
N/S	Out of scope
N/I	Not implemented yet
N/F	Not fully implemented

Below table identifies the AUTOSAR requirements that are not fully implemented, implemented differently, not available, not testable or out of scope for the MEM_43_EXFLS driver.

Requirement	Status	Description	Notes
SWS_Mem_00062	N/S	If the memory hardware provides ECC information, the Mem driver shall check for correctable ECC errors and set the job result code to MEM↵_ECC_CORRECTED and proceed with the current job processing.	Out of scope
SWS_Mem_00063	N/S	If the memory hardware provides ECC information, the Mem driver shall check for uncorrectable ECC errors and set the job result code to MEM↵_ECC_UNCORRECTED and abort the current job processing.	Out of scope
SWS_Mem_00077	N/S	In case the last memory operation was completed but the ECC circuit corrected an ECC error, the job result shall be set to MEM_ECC_↵CORRECTED.	Out of scope
SWS_Mem_00078	N/S	In case the last memory operation didn't complete due to an uncorrectable ECC error, the job result shall be set to MEM_ECC_↵UNCORRECTED.	Out of scope
SWS_Mem_00082	N/I	In case a memory device provides a hardware-based suspend/resume mechanism, the Mem driver shall utilize the hardware capabilities to implement the Mem_Suspend and Mem_Resume services. If no hardware suspend/resume support is available, the Mem_Suspend and Mem_Resume services shall return MEM_E_SERVICE_NOT_AVAIL.	- The update will be implemented later
SWS_Mem_00053	N/I	All hardware specific routines shall be accessed by the Mem_HwSpecific↵Service() service.	-The update will be implemented later
SWS_Mem_00070	N/I	In case a service is not relevant for a specific memory device technology, the service shall always return MEM_E↵_SERVICE_NOT_AVAIL.	-The update will be implemented later
SWS_Mem_00080	N/I	The service Mem_Suspend shall suspend any ongoing flash operations using an according hardware mechanism.	- The update will be implemented later
SWS_Mem_00083	N/I	In case a suspend operation is already in pending, Mem_Suspend shall reject the request by returning E_NOT_OK without further actions.	- The update will be implemented later

Requirement	Status	Description	Notes
SWS_Mem_00085	N/I	If development error detection is enabled by MemGeneral.MemDev↔ErrorDetect, the service Mem_↔Suspend shall check that the Mem driver has been initialized. If this check fails, Mem_Suspend shall raise the development error MEM_E_UNINIT.	-The update will be implemented later
SWS_Mem_00081	N/I	The service Mem_Resume shall resume a flash operations that was suspended by the service Mem_Suspend.	-The update will be implemented later
SWS_Mem_00084	N/I	In case no suspend operation is pending, Mem_Resume shall reject the request by returning E_NOT_OK without further actions.	-The update will be implemented later
SWS_Mem_00086	N/I	If development error detection is enabled by MemGeneral.MemDev↔ErrorDetect, the service Mem_↔Resume shall check that the Mem driver has been initialized. If this check fails, Mem_Resume shall raise the development error MEM_E_UNINIT.	-The update will be implemented later
SWS_Mem_00019	N/I	If development error detection is enabled by MemGeneral.MemDev↔ErrorDetect, the service Mem_↔PropagateError shall check that the Mem driver has been initialized. If this check fails, Mem_Propagate↔Error shall raise the development error MEM_E_UNINIT.	-The update will be implemented later
SWS_Mem_00020	N/I	If development error detection is enabled by MemGeneral.MemDev↔ErrorDetect, the service Mem_↔PropagateError shall check that the provided is consistent with the configuration. If this check fails, Mem_↔PropagateError shall raise the development error MEM_E_PARAM↔INSTANCE_ID.	-The update will be implemented later
SWS_Mem_00061	N/S	If the Mem_PropagateError service is called, the Mem driver shall set the job result code to MEM_ECC_↔UNCORRECTED and cancel the current job processing.	Out of scope
SWS_Mem_00068	N/I	If development error detection is enabled by MemGeneral.MemDev↔ErrorDetect, the service Mem_Hw↔SpecificService shall check that the Mem driver has been initialized. If this check fails, Mem_HwSpecific↔Service shall raise the development error MEM_E_UNINIT.	-The update will be implemented later

Requirement	Status	Description	Notes
SWS_Mem_00026	N/I	If development error detection is enabled by MemGeneral.MemDev↔ErrorDetect, the service Mem_Hw↔SpecificService shall check that the provided is consistent with the configuration. If this check fails, Mem_↔HwSpecificService shall raise the development error MEM_E_PARAM↔_INSTANCE_ID.	-The update will be implemented later
SWS_Mem_00027	N/I	If development error detection is enabled by MemGeneral.MemDev↔ErrorDetect, the service Mem_Hw↔SpecificService shall raise the development error MEM_E_PARAM_↔POINTER if the or argument is a NULL pointer.	-The update will be implemented later

Requirement	Status	Description	Notes
SWS_Mem_10017	N/I	<p>Service Name: Mem_HwSpecific↔</p> <p>Service (draft)</p> <p>Syntax: Std_ReturnType Mem_↔</p> <p>HwSpecificService (</p> <p>Mem_InstanceIdType instanceId,</p> <p>Mem_HwServiceIdType hwServiceId,</p> <p>Mem_DataType* dataPtr,</p> <p>Mem_LengthType* lengthPtr)</p> <p>Service ID [hex]: 0x0a</p> <p>Sync/Async: Asynchronous</p> <p>Reentrancy: Non Reentrant</p> <p>Parameters (in): instanceId: ID of the related memory driver instance.</p> <p>hwServiceId: Hardware specific service request identifier for dispatching the request.</p> <p>dataPtr: Request specific data pointer.</p> <p>lengthPtr: Size pointer of the data passed by dataPtr.</p> <p>Parameters (inout): None</p> <p>Parameters (out): None</p> <p>Return value: Std_ReturnType:</p> <p>E_OK: The requested job has been accepted by the module.</p> <p>E_NOT_OK: The requested job has not been accepted by the module.</p> <p>E_MEM_SERVICE_NOT_AVAIL:</p> <p>The service function is not implemented.</p> <p>Description: Triggers a hardware specific memory driver job. dataPtr can be used to pass and return data to/from this service. This service is just a dispatcher to the hardware specific service implementation referenced by hwServiceId. The result of this service can be retrieved using the Mem_GetJobResult API. If the hardware specific operation was successful, the result of the job is MEM_JOB_OK. If the hardware specific operation failed, the result of the job is MEM_JOB_FAILED.:</p> <p>Tags: atp.Status=draft</p> <p>Available via: Mem.h</p>	-The update will be implemented later

Requirement	Status	Description	Notes
SWS_Mem_10015	N/I	Service Name: Mem_PropagateError (draft) Syntax: void Mem_PropagateError (Mem_InstanceIdType instanceId) Service ID [hex]: 0x08 Sync/Async: Synchronous Reentrancy: Non Reentrant Parameters (in): instanceId: ID of the related memory driver instance. Parameters (inout): None Parameters (out): None Return value: None Description: This service can be used to report an access error in case the Mem driver cannot provide the access error information - typically for ECC faults. It is called by the system ECC handler to propagate an ECC error to the memory upper layers.: Tags← : atp.Status=draft Available via: Mem.h	-The update will be implemented later
SWS_Mem_10025	N/I	Service Name: Mem_Resume (draft) Syntax: void Mem_Resume (Mem_InstanceIdType instanceId) Service ID [hex]: 0x0d Sync/Async: Synchronous Reentrancy: Non Reentrant Parameters (in): instanceId: ID of the related memory driver instance. Parameters (inout): None Parameters (out): None Return value: None Description: Resume suspended memory operation using hardware mechanism.: Tags: atp.Status=draft Available via: Mem.h	-The update will be implemented later

3.5 Driver Limitations

- The following features/APIs are not supported:
 - Suspend/Resume
 - HwSpecificService
 - PropagateError

3.6 Driver usage and configuration tips

None.

3.7 Runtime errors

- There are no runtime errors.

3.7.1 Transient Faults

- The Mem driver does not use transient faults.

3.7.2 Development Errors

- The development error types defined for Mem are listed in the table:

Type of error	Related error	Value (hex)
API service called without module initialization	MEM_E_UNINIT	0x01
API service called with NULL pointer	MEM_E_PARAM_POINTER	0x02
API service called with an invalid address	MEM_E_PARAM_ADDRESS	0x03
API service called with an invalid length	MEM_E_PARAM_LENGTH	0x04
API service called with an invalid driver instance ID	MEM_E_PARAM_INSTANCE_ID	0x05
API service called while a job request is still in progress	MEM_E_JOB_PENDING	0x06

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 [Mem](#)
 - Container [AutosarExt](#)
 - * Parameter [MemEnableUserModeSupport](#)
 - Container [MemGeneral](#)
 - * Parameter [MemDevErrorDetect](#)
 - * Parameter [MemQspiInitCallout](#)
 - * Parameter [MemQspiResetCallout](#)
 - * Parameter [MemQspiErrorCheckCallout](#)
 - * Parameter [MemQspiEccCheckCallout](#)
 - * Parameter [MemExFlsTimeoutMethod](#)
 - * Parameter [MemQspiSyncReadTimeout](#)
 - * Parameter [MemQspiAsyncWriteTimeout](#)
 - * Parameter [MemQspiAsyncEraseTimeout](#)
 - * Parameter [MemQspiSyncWriteTimeout](#)
 - * Parameter [MemQspiSyncEraseTimeout](#)
 - * Parameter [MemQspiDllLockTimeout](#)
 - * Parameter [MemQspiCommandCompleteTimeout](#)
 - * Parameter [MemQspiResetTimeout](#)
 - * Parameter [MemQspiSfpEnableGlobal](#)
 - * Parameter [MemQspiSfpEnableMdad](#)
 - * Parameter [MemQspiSfpEnableFrad](#)
 - * Parameter [MemQspiIdleTimeout](#)
 - * Parameter [MemQspiFlashInitTimeout](#)
 - * Parameter [MemQspiSoftwareResetDelay](#)
 - * Parameter [MemQspiTxBufferResetDelay](#)
 - * Parameter [MemQspiWriteEnableRetries](#)
 - Container [MemInstance](#)
 - * Parameter [MemInstanceId](#)
 - * Reference [MemDeviceRef](#)
 - * Container [MemSectorBatch](#)

- Parameter [MemStartAddress](#)
- Parameter [MemNumberOfSectors](#)
- Parameter [MemEraseSectorSize](#)
- Parameter [MemReadPageSize](#)
- Parameter [MemWritePageSize](#)
- Parameter [MemSpecifiedEraseCycles](#)
- Container [MemBurstSettings](#)
- Parameter [MemEraseBurstSize](#)
- Parameter [MemReadBurstSize](#)
- Parameter [MemWriteBurstSize](#)
- Container [MemDevice](#)
 - * Container [MemCfg](#)
 - Parameter [MemName](#)
 - Parameter [MemAlignment](#)
 - Parameter [MemAHBReadEnable](#)
 - Parameter [MemUseSfdp](#)
 - Parameter [MemConnectionType](#)
 - Reference [MemCfgRef](#)
 - Reference [MemQspiInstance](#)
 - * Container [MemSerialFlashCfg](#)
 - Parameter [MemCfgSize](#)
 - Parameter [MemCfgPageSize](#)
 - Reference [MemCfgReadLUT](#)
 - Reference [MemCfgWriteLUT](#)
 - Reference [MemCfgRead0xxLUT](#)
 - Reference [MemCfgRead0xxLUTAHB](#)
 - Reference [MemCtrlAutoCfgPtr](#)
 - Container [MemCfgReadIdSettings](#)
 - Parameter [MemCfgReadIdSize](#)
 - Parameter [MemQspiDeviceId](#)
 - Reference [MemCfgReadIdLUT](#)
 - Container [MemCfgEraseSettings](#)
 - Parameter [MemCfgErase1Size](#)
 - Parameter [MemCfgErase2Size](#)
 - Parameter [MemCfgErase3Size](#)
 - Parameter [MemCfgErase4Size](#)
 - Reference [MemCfgErase1LUT](#)
 - Reference [MemCfgErase2LUT](#)
 - Reference [MemCfgErase3LUT](#)
 - Reference [MemCfgErase4LUT](#)
 - Reference [MemChipEraseLUT](#)
 - Container [MemStatusConfig](#)
 - Parameter [MemRegSize](#)
 - Parameter [MemBusyOffset](#)
 - Parameter [MemBusyValue](#)
 - Parameter [MemWriteEnableOffset](#)

- Parameter [MemBlockProtectionOffset](#)
- Parameter [MemBlockProtectionWidth](#)
- Parameter [MemBlockProtectionValue](#)
- Reference [MemStatusRegInitReadLut](#)
- Reference [MemStatusRegReadLut](#)
- Reference [MemStatusRegWriteLut](#)
- Reference [MemWriteEnableSRLut](#)
- Reference [MemWriteEnableLut](#)
- Container [MemSuspendSettings](#)
- Reference [MemEraseSuspendLut](#)
- Reference [MemEraseResumeLut](#)
- Reference [MemProgramSuspendLut](#)
- Reference [MemProgramResumeLut](#)
- Container [MemResetSettings](#)
- Parameter [MemResetCmdCount](#)
- Reference [MemResetCmdLut](#)
- Container [MemInitResetSettings](#)
- Parameter [MemResetCmdCount](#)
- Reference [MemResetCmdLut](#)
- Container [MemInitConfiguration](#)
- Parameter [opType](#)
- Parameter [addr](#)
- Parameter [size](#)
- Parameter [shift](#)
- Parameter [width](#)
- Parameter [value](#)
- Reference [MemCommand1Lut](#)
- Reference [MemCommand2Lut](#)
- Reference [MemWeLut](#)
- Reference [MemCtrlCfgPtr](#)
- Container [MemLUT](#)
- Parameter [MemLUTIndex](#)
- Container [MemInstructionOperandPair](#)
- Parameter [MemInstrOperPairIndex](#)
- Parameter [MemLUTInstruction](#)
- Parameter [MemLUTPad](#)
- Parameter [MemLUTOperand](#)
- * Container [MemHyperFlashCfg](#)
 - Parameter [MemCfgSize](#)
 - Parameter [MemCfgPageSize](#)
 - Parameter [MemOutputDriverStrength](#)
 - Parameter [MemRWDSLWOnDualError](#)
 - Parameter [MemSecureRegionUnlocked](#)
 - Parameter [MemReadLatency](#)
 - Parameter [MemParamSectorMap](#)

- Reference [MemCtrlAutoCfgPtr](#)
- Container [MemCfgReadIdSettings](#)
- Parameter [MemCfgReadIdLUT](#)
- Parameter [MemCfgReadIdWordAddr](#)
- Parameter [MemCfgReadIdSize](#)
- Parameter [MemQspiDeviceId](#)
- * Container [MemController](#)
 - Parameter [MemControllerName](#)
 - Reference [MemControllerCfgRef](#)
- * Container [MemControllerCfg](#)
 - Parameter [MemHwUnitReadMode](#)
 - Parameter [MemSerialFlashA1Size](#)
 - Parameter [MemSerialFlashA2Size](#)
 - Parameter [MemHwUnitSamplingModeA](#)
 - Parameter [MemIdleSignalDriveIOFA3HighLvl](#)
 - Parameter [MemIdleSignalDriveIOFA2HighLvl](#)
 - Parameter [MemHwUnitSamplingEdge](#)
 - Parameter [MemHwUnitSamplingDly](#)
 - Parameter [MemHwUnitTdh](#)
 - Parameter [MemHwUnitTcsh](#)
 - Parameter [MemHwUnitTcss](#)
 - Parameter [MemHwUnitColumnAddressWidth](#)
 - Parameter [MemHwUnitByteSwapping](#)
 - Parameter [MemHwUnitWordAddressable](#)
 - Container [MemAhbBuffer](#)
 - Parameter [MemAhbBufferInstance](#)
 - Parameter [MemAhbBufferMasterId](#)
 - Parameter [MemAhbBufferSize](#)
 - Parameter [MemAhbBufferAllMasters](#)
 - Container [MemDllCfgA](#)
 - Parameter [MemDllCfgADllMode](#)
 - Parameter [MemDllCfgADllCraFreqEn](#)
 - Parameter [MemDllCfgADllCraReferenceCounter](#)
 - Parameter [MemDllCfgADllCraResolution](#)
 - Parameter [MemDllCfgADllCraSlvFineOffset](#)
 - Parameter [MemDllCfgADllCraSlvDlyOffset](#)
 - Parameter [MemDllCfgADllCraSlvDlyCoarse](#)
 - Parameter [MemDllCfgADllTapSelect](#)
 - Container [MemSecureFlashProtection](#)
 - Parameter [MemQspiSfpMasterTimeout](#)
 - Container [MemQspiSfpMdadTG](#)

- Parameter [Valid](#)
- Parameter [SecureAttribute](#)
- Parameter [MaskType](#)
- Parameter [Mask](#)
- Parameter [DomainID](#)
- Container [MemQspiSfpFrad](#)
- Parameter [Valid](#)
- Parameter [StartAddress](#)
- Parameter [EndAddress](#)
- Parameter [ExclusiveAccessLock](#)
- Parameter [ExclusiveAccessOwner](#)
- Parameter [Md0Acp](#)
- Parameter [Md1Acp](#)
- Container [MemPublishedInformation](#)
 - * Parameter [MemErasedValue](#)
- Container [CommonPublishedInformation](#)
 - * Parameter [ArReleaseMajorVersion](#)
 - * Parameter [ArReleaseMinorVersion](#)
 - * Parameter [ArReleaseRevisionVersion](#)
 - * Parameter [ModuleId](#)
 - * Parameter [SwMajorVersion](#)
 - * Parameter [SwMinorVersion](#)
 - * Parameter [SwPatchVersion](#)
 - * Parameter [VendorApiInfix](#)
 - * Parameter [VendorId](#)

4.1 Module Mem

Configuration of the Mem_43_ExFls module.

Included containers:

- [AutosarExt](#)
- [MemGeneral](#)
- [MemInstance](#)
- [MemDevice](#)
- [MemPublishedInformation](#)
- [CommonPublishedInformation](#)

Property	Value
type	ECUC-MODULE-DEF
lowerMultiplicity	1
upperMultiplicity	Infinite
postBuildVariantSupport	false
supportedConfigVariants	VARIANT-PRE-COMPILE

4.2 Container AutosarExt

Vendor specific: This container contains the global Non-Autosar configuration parameters of the Mem 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

4.3 Parameter MemEnableUserModeSupport

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

configuring REG_PROT for 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
symbolicNameValue	false
lowerMultiplicity	1

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

4.4 Container MemGeneral

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

4.5 Parameter MemDevErrorDetect

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-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.6 Parameter MemQspiInitCallout

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.

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

4.7 Parameter MemQspiResetCallout

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 the possibility of resetting the external memory to an idle and error free state.

If the callout is disabled, at the beginning of a new job the 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 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.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	MemQspiResetCallout

4.8 Parameter MemQspiErrorCheckCallout

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.

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

4.9 Parameter MemQspiEccCheckCallout

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.

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

4.10 Parameter MemExFlsTimeoutMethod

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-PRE-COMPILE: PRE-COMPILE
defaultValue	OSIF_COUNTER_DUMMY
literals	['OSIF_COUNTER_DUMMY', 'OSIF_COUNTER_SYSTEM', 'OSIF_COUNTER_CUSTOM']

4.11 Parameter MemQspiSyncReadTimeout

Vendor specific: Mem 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-PRE-COMPILE: PRE-COMPILE
defaultValue	10000000
max	2147483647
min	0

4.12 Parameter MemQspiAsyncWriteTimeout

Vendor specific: Mem 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-PRE-COMPILE: PRE-COMPILE
defaultValue	10000000
max	2147483647
min	0

4.13 Parameter MemQspiAsyncEraseTimeout

Vendor specific: Mem 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-PRE-COMPILE: PRE-COMPILE
defaultValue	10000000
max	2147483647
min	0

4.14 Parameter MemQspiSyncWriteTimeout

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

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-PRE-COMPILE: PRE-COMPILE
defaultValue	10000000
max	2147483647
min	0

4.15 Parameter MemQspiSyncEraseTimeout

Vendor specific: Mem 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-PRE-COMPILE: PRE-COMPILE
defaultValue	10000000
max	2147483647
min	0

4.16 Parameter MemQspiDllLockTimeout

Vendor specific: Mem Qspi DLL Lock Timeout is the timeout value for waiting DLL lock bit.

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

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	10000000
max	2147483647
min	0

4.17 Parameter MemQspiCommandCompleteTimeout

Vendor specific: Mem 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-PRE-COMPILE: PRE-COMPILE
default Value	10000000
max	2147483647
min	0

4.18 Parameter MemQspiResetTimeout

Vendor specific: Mem 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

Property	Value
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	10000000
max	2147483647
min	0

4.19 Parameter MemQspiSfpEnableGlobal

Vendor specific: Master Global Configuration (MGC) :: Global Valid [GVLD]

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-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.20 Parameter MemQspiSfpEnableMdad

Vendor specific: Global Valid MDAD [GVLDMDAD]

Checked: MDAD checks are enabled.

Unchecked: MDAD checks are 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-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.21 Parameter MemQspiSfpEnableFrad

Vendor specific: Global Valid FRAD [GVLDFRAD]

Checked: FRAD checks are enabled.

Unchecked: FRAD checks are 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-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.22 Parameter MemQspiIdleTimeout

Vendor specific: How much time to wait for the QSPI to become idle.

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-PRE-COMPILE: PRE-COMPILE
defaultValue	100
max	2147483647
min	0

4.23 Parameter MemQspiFlashInitTimeout

Vendor specific: Fls Flash Init Timeout is the timeout for completing the initialization 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-PRE-COMPILE: PRE-COMPILE
default Value	1
max	2147483647
min	0

4.24 Parameter MemQspiSoftwareResetDelay

Vendor specific: Mem 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-PRE-COMPILE: PRE-COMPILE
default Value	21
max	2147483647
min	0

4.25 Parameter MemQspiTxBufferResetDelay

Vendor specific: Mem 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-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	2147483647
min	0

4.26 Parameter MemQspiWriteEnableRetries

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-PRE-COMPILE: PRE-COMPILE
defaultValue	3
max	100
min	0

4.27 Container MemInstance

This container includes the Mem driver instance specific configuration parameters.

Included subcontainers:

- [MemSectorBatch](#)

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

4.28 Parameter MemInstanceId

The unique numeric identifier which is used to reference a Mem driver instance in case multiple devices of the same type shall be addressed by one Mem driver.

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-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	65535
min	0

4.29 Reference MemDeviceRef

Vendor specific: Memory device instance to which this memory instance belongs.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemCfg

4.30 Container MemSectorBatch

Configuration description of a programmable sector or sector batch.

Sector batch means that homogenous and coherent sectors can be configured as one MemSector element.

It is recommended to group as many identical sectors as possible together.

Included subcontainers:

- [MemBurstSettings](#)

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

4.31 Parameter MemStartAddress

Physical start address of the sector (batch).

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

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	9223372036854775807
min	0

4.32 Parameter MemNumberOfSectors

Number of contiguous sectors with identical values for MemSectorSize and MemPageSize.

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-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	65535
min	1

4.33 Parameter MemEraseSectorSize

Size of this sector in bytes.

A sector is the smallest erasable unit.

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

Property	Value
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	4096
max	4294967295
min	1

4.34 Parameter MemReadPageSize

Size of a read page of this sector in bytes.

A read page is the smallest readable unit.

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-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	4294967295
min	1

4.35 Parameter MemWritePageSize

Size of a write page of this sector in bytes.

A write page is the smallest writeable unit.

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

Property	Value
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	8
max	4294967295
min	1

4.36 Parameter MemSpecifiedEraseCycles

Number of erase cycles specified for the memory device

(usually given in the device data sheet).

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-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.37 Container MemBurstSettings

Container for burst setting configuration parameters of the Mem driver.

A sector burst can be used for improved performance.

Included subcontainers:

- None

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

4.38 Parameter MemEraseBurstSize

Size of sector erase burst in bytes. A sector burst can be used for improved performance and is typically (a subset of) a sector batch.

To make use of the sector erase burst feature, the physical start address of the sector batch must be aligned to the sector erase burst size.

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-PRE-COMPILE: PRE-COMPILE
defaultValue	4096
max	4294967295
min	1

4.39 Parameter MemReadBurstSize

This value specifies the maximum number of bytes the MemAcc module requests within one Mem read request.

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-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	4294967295
min	1

4.40 Parameter MemWriteBurstSize

Size of page write/program burst in bytes. A sector burst can be used

for improved performance and is typically (a subset of) a sector batch.

To make use of the write burst feature, the physical start address must

be aligned to the write burst size.

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-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	4294967295
min	1

4.41 Container MemDevice

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

Included subcontainers:

- [MemCfg](#)
- [MemSerialFlashCfg](#)
- [MemHyperFlashCfg](#)
- [MemController](#)
- [MemControllerCfg](#)

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

4.42 Container MemCfg

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

Included subcontainers:

- None

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

4.43 Parameter MemName

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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	Device_0

4.44 Parameter MemAlignment

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

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
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	1
max	16
min	1

4.45 Parameter MemAHBReadEnable

Vendor specific: When set, the driver will configure the related AHB settings 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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	false

4.46 Parameter MemUseSfdp

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
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.47 Parameter MemConnectionType

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	false
valueConfigClasses	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']

4.48 Reference MemCfgRef

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

Property	Value
type	ECUC-CHOICE-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destinations	['/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg', '/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemHyper↔ FlashCfg']

4.49 Reference MemQspiInstance

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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemController

4.50 Container MemSerialFlashCfg

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

Included subcontainers:

- [MemCfgReadIdSettings](#)
- [MemCfgEraseSettings](#)
- [MemStatusConfig](#)
- [MemSuspendSettings](#)
- [MemResetSettings](#)
- [MemInitResetSettings](#)
- [MemInitConfiguration](#)
- [MemLUT](#)

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

4.51 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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	8388608
max	4294967295
min	0

4.52 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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	256
max	4294967295
min	0

4.53 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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.54 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

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.55 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-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.56 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-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false

Property	Value
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.57 Reference MemCtrlAutoCfgPtr

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-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemControllerCfg

4.58 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

4.59 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
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	3
max	10
min	0

4.60 Parameter MemQspiDeviceId

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 MemQspiDeviceId = 0x3A:81:C2, 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-STRING-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0x3A:81:C2

4.61 Reference MemCfgReadIdLUT

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

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.62 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

4.63 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
origin	NXP

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

4.64 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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	12
max	32
min	1

4.65 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
lowerMultiplicity	1
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	12
max	32
min	1

4.66 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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	12
max	32
min	1

4.67 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
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE

Property	Value
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.68 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-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.69 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-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.70 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-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.71 Reference MemChipEraseLUT

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-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.72 Container MemStatusConfig

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

4.73 Parameter MemRegSize

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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	4
min	1

4.74 Parameter MemBusyOffset

Vendor specific: Position of "busy" bit inside status register. This bit 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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE

Property	Value
defaultValue	0
max	31
min	0

4.75 Parameter MemBusyValue

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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	1
min	0

4.76 Parameter MemWriteEnableOffset

Vendor specific: Position of "write enable" bit 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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	31
min	0

4.77 Parameter MemBlockProtectionOffset

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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	2
max	31
min	0

4.78 Parameter MemBlockProtectionWidth

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

4.79 Parameter MemBlockProtectionValue

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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	0
max	15
min	0

4.80 Reference MemStatusRegInitReadLut

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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.81 Reference MemStatusRegReadLut

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

Property	Value
type	ECUC-REFERENCE-DEF

Property	Value
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.82 Reference MemStatusRegWriteLut

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
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.83 Reference MemWriteEnableSRLut

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

Property	Value
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.84 Reference MemWriteEnableLut

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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.85 Container MemSuspendSettings

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

4.86 Reference MemEraseSuspendLut

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-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.87 Reference MemEraseResumeLut

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-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.88 Reference MemProgramSuspendLut

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

Property	Value
type	ECUC-REFERENCE-DEF

Property	Value
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.89 Reference MemProgramResumeLut

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-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.90 Container MemResetSettings

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	false
multiplicityConfigClasses	N/A

4.91 Parameter MemResetCmdCount

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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	255
min	1

4.92 Reference MemResetCmdLut

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-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.93 Container MemInitResetSettings

Vendor specific: Container related to software reset command, for resetting the flash device. This reset procedure applies only at driver initialization. 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

4.94 Parameter MemResetCmdCount

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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	255
min	1

4.95 Reference MemResetCmdLut

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-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE

Property	Value
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.96 Container MemInitConfiguration

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

4.97 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

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_OP_TYPE_CMD
literals	['QSPI_IP_OP_TYPE_CMD', 'QSPI_IP_OP_TYPE_WRITE_REG', 'QSPI_IP_OP_TYPE_RMW_REG', 'QSPI_IP_OP_TYPE_READ_REG', 'QSPI_IP_OP_TYPE_QSPI_CFG']

4.98 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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.99 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

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

4.100 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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	32
min	0

4.101 Parameter width

Vendor specific: Width 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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE

Property	Value
defaultValue	0
max	32
min	0

4.102 Parameter value

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

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-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	4294967295
min	0

4.103 Reference MemCommand1Lut

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-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.104 Reference MemCommand2Lut

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
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.105 Reference MemWeLut

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-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemSerialFlash↔ Cfg/MemLUT

4.106 Reference MemCtrlCfgPtr

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
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemControllerCfg

4.107 Container MemLUT

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 occupy up to 4 LUTs,

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

Included subcontainers:

- [MemInstructionOperandPair](#)

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

4.108 Parameter MemLUTIndex

Vendor specific: Mem 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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	65534
min	0

4.109 Container MemInstructionOperandPair

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

4.110 Parameter MemInstrOperPairIndex

Vendor specific: Memory 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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	65534
min	0

4.111 Parameter MemLUTInstruction

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
postBuildVariantValue	false

Property	Value
valueConfigClasses	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_LUT_INSTR_READ', 'QSPI_IP_LUT_INSTR_WRITE', 'QSPI_IP_LUT_INSTR_JMP_ON_CS', 'QSPI_IP_LUT_INSTR_ADDR_DDR', 'QSPI_IP_LUT_INSTR_MODE_DDR', 'QSPI_IP_LUT_INSTR_MODE2_DDR', 'QSPI_IP_LUT_INSTR_MODE4_DDR', 'QSPI_IP_LUT_INSTR_READ_DDR', 'QSPI_IP_LUT_INSTR_WRITE_DDR', 'QSPI_IP_LUT_INSTR_DATA_LEARN', 'QSPI_IP_LUT_INSTR_CMD_DDR', 'QSPI_IP_LUT_INSTR_CADDR', 'QSPI_IP_LUT_INSTR_CADDR_DDR', 'QSPI_IP_LUT_INSTR_JMP_TO_SEQ', 'QSPI_IP_LUT_INSTR_STOP']

4.112 Parameter MemLUTPad

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	false
valueConfigClasses	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']

4.113 Parameter MemLUTOperand

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

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-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	255
min	0

4.114 Container MemHyperFlashCfg

Container for defining configurations for hyper flash devices.

These configurations are not tied to a particular device instance.

Included subcontainers:

- [MemCfgReadIdSettings](#)

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

4.115 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

Property	Value
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	8388608
max	4294967295
min	0

4.116 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-INTEGGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	512
max	4294967295
min	0

4.117 Parameter MemOutputDriverStrength

Output driver level of the device:

QSPI_IP_HF_DRV_STRENGTH_000 : Typical Impedance for 1.8V: 27; Typical Impedance 3V: 20.

QSPI_IP_HF_DRV_STRENGTH_001 : Typical Impedance for 1.8V: 117; Typical Impedance 3V: 71.

QSPI_IP_HF_DRV_STRENGTH_002 : Typical Impedance for 1.8V: 68; Typical Impedance 3V: 40.

QSPI_IP_HF_DRV_STRENGTH_003 : Typical Impedance for 1.8V: 45; Typical Impedance 3V: 27.

QSPI_IP_HF_DRV_STRENGTH_004 : Typical Impedance for 1.8V: 34; Typical Impedance 3V: 20.

QSPI_IP_HF_DRV_STRENGTH_005 : Typical Impedance for 1.8V: 27; Typical Impedance 3V: 16.

QSPI_IP_HF_DRV_STRENGTH_006 : Typical Impedance for 1.8V: 24; Typical Impedance 3V: 14.

QSPI_IP_HF_DRV_STRENGTH_007 : Typical Impedance for 1.8V: 20; Typical Impedance 3V: 12.

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-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_HF_DRV_STRENGTH_000
literals	['QSPI_IP_HF_DRV_STRENGTH_000', 'QSPI_IP_HF_DRV_STRENGTH_001', 'QSPI_IP_HF_DRV_STRENGTH_002', 'QSPI_IP_HF_DRV_STRENGTH_003', 'QSPI_IP_HF_DRV_STRENGTH_004', 'QSPI_IP_HF_DRV_STRENGTH_005', 'QSPI_IP_HF_DRV_STRENGTH_006', 'QSPI_IP_HF_DRV_STRENGTH_007']

4.118 Parameter MemRWDSLlowOnDualError

Specifies if RWDS will stall upon Dual Error Detect

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-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.119 Parameter MemSecureRegionUnlocked

If true, the secure silicon region will be unlocked

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-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.120 Parameter MemReadLatency

Read latency in cycles. Must be set considering the operation frequency:

QSPI_IP_HF_READ_LATENCY_5_CLOCKS : Read latency 5 clocks; max frequency : 52 MHz

QSPI_IP_HF_READ_LATENCY_6_CLOCKS : Read latency 6 clocks; max frequency : 62 MHz

QSPI_IP_HF_READ_LATENCY_7_CLOCKS : Read latency 7 clocks; max frequency : 72 MHz

QSPI_IP_HF_READ_LATENCY_8_CLOCKS : Read latency 8 clocks; max frequency : 83 MHz

QSPI_IP_HF_READ_LATENCY_9_CLOCKS : Read latency 9 clocks; max frequency : 93 MHz

QSPI_IP_HF_READ_LATENCY_10_CLOCKS : Read latency 10 clocks; max frequency : 104 MHz

QSPI_IP_HF_READ_LATENCY_11_CLOCKS : Read latency 11 clocks; max frequency : 114 MHz

QSPI_IP_HF_READ_LATENCY_12_CLOCKS : Read latency 12 clocks; max frequency : 125 MHz

QSPI_IP_HF_READ_LATENCY_13_CLOCKS : Read latency 13 clocks; max frequency : 135 MHz

QSPI_IP_HF_READ_LATENCY_14_CLOCKS : Read latency 14 clocks; max frequency : 145 MHz

QSPI_IP_HF_READ_LATENCY_15_CLOCKS : Read latency 15 clocks; max frequency : 156 MHz

QSPI_IP_HF_READ_LATENCY_16_CLOCKS : Read latency 16 clocks; max frequency : 166 MHz

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

Property	Value
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_HF_READ_LATENCY_16_CLOCKS
literals	['QSPI_IP_HF_READ_LATENCY_5_CLOCKS', 'QSPI_IP_HF_READ_↵ LATENCY_6_CLOCKS', 'QSPI_IP_HF_READ_LATENCY_7_CLOCKS', 'QSPI_IP_HF_READ_LATENCY_8_CLOCKS', 'QSPI_IP_HF_READ_↵ LATENCY_9_CLOCKS', 'QSPI_IP_HF_READ_LATENCY_10_CLOCKS', 'QSPI_IP_HF_READ_LATENCY_11_CLOCKS', 'QSPI_IP_HF_READ_↵ LATENCY_12_CLOCKS', 'QSPI_IP_HF_READ_LATENCY_13_↵ CLOCKS', 'QSPI_IP_HF_READ_LATENCY_14_CLOCKS', 'QSPI_IP_↵ HF_READ_LATENCY_15_CLOCKS', 'QSPI_IP_HF_READ_LATENCY_↵ _16_CLOCKS']

4.121 Parameter MemParamSectorMap

Define the mapping of the 4-KB Parameter Sectors:

QSPI_IP_HF_PARAM_AND_PASSWORD_MAP_LOW : Parameter-Sectors and Read Password Sectors mapped into lowest addresses

QSPI_IP_HF_PARAM_AND_PASSWORD_MAP_HIGH : Parameter-Sectors and Read Password Sectors mapped into highest addresses

QSPI_IP_HF_UNIFORM_SECTORS_READ_PASSWORD_LOW : Uniform Sectors with Read Password Sector mapped into lowest addresses

QSPI_IP_HF_UNIFORM_SECTORS_READ_PASSWORD_HIGH : Uniform Sectors with Read Password Sector mapped into highest addresses

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-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_HF_UNIFORM_SECTORS_READ_PASSWORD_LOW
literals	['QSPI_IP_HF_PARAM_AND_PASSWORD_MAP_LOW', 'QSPI_IP_HF_↵ _PARAM_AND_PASSWORD_MAP_HIGH', 'QSPI_IP_HF_UNIFORM_↵ _SECTORS_READ_PASSWORD_LOW', 'QSPI_IP_HF_UNIFORM_↵ SECTORS_READ_PASSWORD_HIGH']

4.122 Reference MemCtrlAutoCfgPtr

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-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemControllerCfg

4.123 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

4.124 Parameter MemCfgReadIdLUT

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

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	False
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_HF_LUT_READ
literals	['QSPI_IP_HF_LUT_READ']

4.125 Parameter MemCfgReadIdWordAddr

Vendor specific: The word address of the device ID in ASO.

This value will be converted to by address for the read ID transaction.

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-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.126 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

Property	Value
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	3
max	10
min	0

4.127 Parameter MemQspiDeviceId

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 MemQspiDeviceId = 0x3A:81:C2, 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-STRING-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0x3A:81:C2

4.128 Container MemController

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

Included subcontainers:

- None

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

4.129 Parameter MemControllerName

Vendor specific: The name of the configured hardware 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
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_INSTANCE_0
literals	['QSPI_IP_INSTANCE_0']

4.130 Reference MemControllerCfgRef

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

Note: This option 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.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_ExFls_TS_T40D34M30I0R0/Mem/MemDevice/MemControllerCfg

4.131 Container MemControllerCfg

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

Included subcontainers:

- [MemAhbBuffer](#)
- [MemDllCfgA](#)
- [MemSecureFlashProtection](#)

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

4.132 Parameter MemHwUnitReadMode

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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_DATA_RATE_SDR
literals	['QSPI_IP_DATA_RATE_SDR', 'QSPI_IP_DATA_RATE_DDR']

4.133 Parameter MemSerialFlashA1Size

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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.134 Parameter MemSerialFlashA2Size

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
lowerMultiplicity	1
upperMultiplicity	1

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

4.135 Parameter MemHwUnitSamplingModeA

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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_READ_MODE_LOOPBACK
literals	['QSPI_IP_READ_MODE_LOOPBACK', 'QSPI_IP_READ_MODE_↵ EXTERNAL_DQS']

4.136 Parameter MemIdleSignalDriveIOFA3HighLvl

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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	true

4.137 Parameter MemIdleSignalDriveIOFA2HighLvl

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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	true

4.138 Parameter MemHwUnitSamplingEdge

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

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_SAMPLE_PHASE_NON_INVERTED
literals	['QSPI_IP_SAMPLE_PHASE_NON_INVERTED', 'QSPI_IP_SAMPLE_↔ PHASE_INVERTED']

4.139 Parameter MemHwUnitSamplingDly

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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_SAMPLE_DELAY_SAME_DQS
literals	['QSPI_IP_SAMPLE_DELAY_SAME_DQS', 'QSPI_IP_SAMPLE_↔ DELAY_HALFCYCLE_EARLY_DQS']

4.140 Parameter MemHwUnitTdh

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 Quad-SPI.

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	false
valueConfigClasses	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']

4.141 Parameter MemHwUnitTcsh

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:

$TCSH = 1 \text{ SCK clk if } N = 0/1 \text{ else, } N \text{ SCK clk if } N > 1$, 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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	3
max	15
min	0

4.142 Parameter MemHwUnitTcss

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:

$TCSS = 0.5 \text{ SCK clk if } N = 0/1 \text{ else, } N + 0.5 \text{ SCK clk if } N > 1$, 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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	3
max	15
min	0

4.143 Parameter MemHwUnitColumnAddressWidth

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

Example: If the coulumn 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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	15
min	0

4.144 Parameter MemHwUnitByteSwapping

Vendor specific: In case of Octal DDR mode, this bit controls whether a word unit composed of two bytes from posedge

and negedge of a single DQS cycle needs to be swapped.

- Disable : One word of two bytes at [nth, n+1th] address.

- Enable : One word of two bytes at [n+1th, nth] address

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-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.145 Parameter MemHwUnitWordAddressable

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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.146 Container MemAhbBuffer

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

4.147 Parameter MemAhbBufferInstance

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 available 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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	AHB_BUFFER_1
literals	['AHB_BUFFER_0', 'AHB_BUFFER_1', 'AHB_BUFFER_2', 'AHB_BUFFER_3']

4.148 Parameter MemAhbBufferMasterId

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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	15
min	0

4.149 Parameter MemAhbBufferSize

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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.150 Parameter MemAhbBufferAllMasters

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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.151 Container MemDllCfgA

Vendor specific: Container for DLL settings for side A

Included subcontainers:

- None

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

4.152 Parameter MemDllCfgADllMode

Vendor specific: Choose the mode of DLL feature for flash A.

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-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_DLL_BYPASSED
literals	['QSPI_IP_DLL_BYPASSED', 'QSPI_IP_DLL_MANUAL_UPDATE', 'QSPI_IP_DLL_AUTO_UPDATE']

4.153 Parameter MemDllCfgADllCraFreqEn

Vendor specific: Frequency enable for flash A. These are 60-133Mhz (low freq) and 133-200Mhz (high freq)

Disable - Selects delay-chain for low frequency of operation.

Enable - Selects delay-chain for high frequency of operation.

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-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.154 Parameter MemDllCfgADllCraReferenceCounter

Vendor specific: Select the "n+1" interval of DLL phase detection and reference delay updating interval (minimum recommended value = 1).

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-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	15
min	0

4.155 Parameter MemDllCfgADllCraResolution

Vendor specific: Minimum resolution for DLL phase detector to remain locked/unlocked based on flash memory clock jitter.

The minimum value is 2, and should be programmed to a more suitable value, such as 6.

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-PRE-COMPILE: PRE-COMPILE
defaultValue	2
max	15
min	0

4.156 Parameter MemDllCfgADllCraSlvFineOffset

Vendor specific: Fine offset delay elements in incoming DQS for flash A

This field sets the number of fine offset delay elements up to 16 in incoming DQS; default should be 1 element

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-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	15
min	0

4.157 Parameter MemDllCfgADllCraSlvDlyOffset

Vendor specific: T/16 offset delay elements in incoming DQS for flash A

This field sets the number of T/16 offset delay elements in incoming DQS; default is 0.

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-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	7
min	0

4.158 Parameter MemDllCfgADllCraSlvDlyCoarse

Vendor specific: Delay elements in each delay tap for flash A

This field sets the number of delay elements in each delay tap. The field is used to overwrite DLL generated delay values and works in BYPASS 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-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	15
min	0

4.159 Parameter MemDllCfgADllTapSelect

Vendor specific: Selects the nth tap provided by slave delay chain for flash memory A.

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-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	7
min	0

4.160 Container MemSecureFlashProtection

Container for configuring the SFP block.

Included subcontainers:

- [MemQspiSfpMdadTG](#)
- [MemQspiSfpFrad](#)

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	N/A

4.161 Parameter MemQspiSfpMasterTimeout

Vendor specific: Master Timeout (MTO)

Maximum timeout value to abort the ongoing write or read command. The timeout counter starts after the access from any target queue has won the arbitration and QSPI is IDLE (FSM_STAT state field is not 00).

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-PRE-COMPILE: PRE-COMPILE
defaultValue	65535
max	4294967295
min	0

4.162 Container MemQspiSfpMdadTG

Target Group n Master Domain Access Descriptor (TG0MDAD - TG1MDAD).

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	2
upperMultiplicity	2
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.163 Parameter Valid

Indicates whether MDAD Descriptor for the target group n is valid.

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-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.164 Parameter SecureAttribute

Configure the SecureAttribute field for this TG.

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-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_SFP_BOTH
literals	['QSPI_IP_SFP_UNSECURE', 'QSPI_IP_SFP_SECURE', 'QSPI_IP_SFP↔_BOTH']

4.165 Parameter MaskType

Vendor specific: Choose the mode of DLL feature for flash A.

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-PRE-COMPILE: PRE-COMPILE
default Value	QSPI_IP_SFP_MASK_AND
literals	['QSPI_IP_SFP_MASK_AND', 'QSPI_IP_SFP_MASK_OR']

4.166 Parameter Mask

Defines the 6-bit mask value for the ID-Match comparison.

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-PRE-COMPILE: PRE-COMPILE
default Value	0
max	63
min	0

4.167 Parameter DomainID

Specifies the reference value of the Domain-ID (MID) for MID-comparison.

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

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	0
max	63
min	0

4.168 Container MemQspiSfpFrad

Flash Region Access Descriptors.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	8
upperMultiplicity	8
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.169 Parameter Valid

Indicates whether the FRAD Descriptor is valid.

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-PRE-COMPILE: PRE-COMPILE
default Value	false

4.170 Parameter StartAddress

Flash Region Start Address (FRAD0_WORD0 - FRAD7_WORD0)

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-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.171 Parameter EndAddress

Flash Region End Address (FRAD0_WORD1 - FRAD7_WORD1)

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-PRE-COMPILE: PRE-COMPILE
defaultValue	4294967295
max	4294967295
min	0

4.172 Parameter ExclusiveAccessLock

Defines the exclusive access lock field of the corresponding FRAD.

QSPI_IP_SFP_EAL_DISABLED: Write permissions available for all masters.

QSPI_IP_SFP_EAL_NONE: Lock enabled. Write permissions revoked for all domains.

QSPI_IP_SFP_EAL_OWNER: Lock enabled. Exclusive write permission for master with domain ID given in ExclusiveAccessOwner based on its Access Control Policy (Md0Acp and Md1Acp). Write disabled for other masters.

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-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_SFP_EAL_DISABLED
literals	['QSPI_IP_SFP_EAL_DISABLED', 'QSPI_IP_SFP_EAL_NONE', 'QSPI_IP_SFP_EAL_OWNER']

4.173 Parameter ExclusiveAccessOwner

Indicates the domain/master ID that owns the exclusive access when ExclusiveAccessLock is QSPI_IP_SFP_EAL_OWNER.

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-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	63
min	0

4.174 Parameter Md0Acp

Master Domain Access Control Policy for transactions coming through TG0.

Value

Policy

Secure privilege

Secure user

Non secure privilege

Non secure user

7

PRIVILEGED

R/W

R

R/W

R

6

ALL

R/W

R/W

R/W

R/W

5

SECURE_PRIVILEGED

R/W

R

R

R

4

SECURE

R/W

R/W

R

R

0

NONE

R

R

R

R

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-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_SFP_ACP_NONE
literals	['QSPI_IP_SFP_ACP_NONE', 'QSPI_IP_SFP_ACP_SECURE', 'QSPI_IP_SFP_ACP_SECURE_PRIVILEGED', 'QSPI_IP_SFP_ACP_ALL', 'QSPI_IP_SFP_ACP_PRIVILEGED']

4.175 Parameter Md1Acp

Master Domain Access Control Policy for transactions coming through TG1.

Value

Policy

Secure privilege

Secure user

Non secure privilege

Non secure user

7

PRIVILEGED

R/W

R

R/W

R

6

ALL

R/W

R/W

R/W

R/W

5

SECURE_PRIVILEGED

R/W

R

R

R

4

SECURE

R/W

R/W

R

R

0

NONE

R

R

R

R

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-PRE-COMPILE: PRE-COMPILE
defaultValue	QSPI_IP_SFP_ACP_NONE
NXP Semiconductors	QSPI_IP_SFP_ACP_NONE
literals	['QSPI_IP_SFP_ACP_NONE', 'QSPI_IP_SFP_ACP_SECURE', 'QSPI_IP_SFP_ACP_SECURE_PRIVILEGED', 'QSPI_IP_SFP_ACP_ALL',

4.176 Container MemPublishedInformation

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

4.177 Parameter MemErasedValue

The contents of an erased 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-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	4294967295
max	4294967295
min	0

4.178 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

4.179 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-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	4
max	4
min	4

4.180 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
postBuildVariantValue	false

Property	Value
valueConfigClasses	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	7
max	7
min	7

4.181 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-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.182 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
valueConfigClasses	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	91
max	91
min	91

4.183 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-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	3
max	3
min	3

4.184 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
valueConfigClasses	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.185 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-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.186 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-PRE-COMPILE: PUBLISHED-INFORMATION
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	

4.187 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-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:

MEM_43_EXFLS Driver	144
QSPI IPV Driver	155

Chapter 6

Module Documentation

6.1 MEM_43_EXFLS Driver

6.1.1 Detailed Description

Data Structures

- struct [Mem_43_ExFls_MemDeviceType](#)
Mem device config type. [More...](#)
- struct [Mem_43_ExFls_SectorBatchType](#)
Sector Batch Type. [More...](#)
- struct [Mem_43_ExFls_MemInstanceType](#)
Mem Instance Type. [More...](#)
- struct [Mem_43_ExFls_ConfigType](#)
Mem Configuration Type. [More...](#)
- struct [Mem_43_ExFls_JobRuntimeInfoType](#)
Mem job runtime information Type. [More...](#)

Macros

- `#define MEM_43_EXFLS_MODULE_ID`
AUTOSAR module identification.
- `#define MEM_43_EXFLS_INSTANCE_ID`
AUTOSAR module instance identification.
- `#define MEM_43_EXFLS_E_UNINIT`
Development error codes (passed to DET)
- `#define MEM_43_EXFLS_DEINIT_ID`
All service IDs (passed to DET)

Types Reference

- typedef uint32 [Mem_43_ExFls_AddressType](#)
Mem Address Type.
- typedef uint32 [Mem_43_ExFls_InstanceIdType](#)
Mem Instance Id Type.
- typedef uint32 [Mem_43_ExFls_LengthType](#)
Mem Length Type.
- typedef uint8 [Mem_43_ExFls_DataType](#)
Mem Data Type.
- typedef uint16 [Mem_43_ExFls_CrcType](#)
Mem CRC Type.
- typedef uint32 [Mem_43_ExFls_HwServiceIdType](#)
Mem Hardware Service Id Type.

Enum Reference

- enum [Mem_43_ExFls_JobResultType](#)
Mem job result type.
- enum [Mem_43_ExFls_JobType](#)
Type of job currently executed by Mem_43_ExFls_MainFunction.

Function Reference

- void [Mem_43_ExFls_Init](#) (const [Mem_43_ExFls_ConfigType](#) *ConfigPtr)
The function initializes Mem_43_ExFls module.
- void [Mem_43_ExFls_DeInit](#) (void)
The function de-initializes the Mem_43_ExFls module.
- void [Mem_43_ExFls_GetVersionInfo](#) (Std_VersionInfoType *VersionInfoPtr)
Return the version information of the Mem module.
- [Mem_43_ExFls_JobResultType](#) [Mem_43_ExFls_GetJobResult](#) ([Mem_43_ExFls_InstanceIdType](#) InstanceId)
Returns the result of the most recent job.
- void [Mem_43_ExFls_Suspend](#) ([Mem_43_ExFls_InstanceIdType](#) InstanceId)
Suspends active memory operation using hardware mechanism.
- void [Mem_43_ExFls_Resume](#) ([Mem_43_ExFls_InstanceIdType](#) InstanceId)
Resumes suspended memory operation using hardware mechanism.
- void [Mem_43_ExFls_PropagateError](#) ([Mem_43_ExFls_InstanceIdType](#) InstanceId)
Propagates an ECC error to the memory upper layers.
- Std_ReturnType [Mem_43_ExFls_Read](#) ([Mem_43_ExFls_InstanceIdType](#) InstanceId, [Mem_43_ExFls_AddressType](#) SourceAddress, [Mem_43_ExFls_DataType](#) *DestinationDataPtr, [Mem_43_ExFls_LengthType](#) Length)
Reads from flash memory.
- Std_ReturnType [Mem_43_ExFls_Write](#) ([Mem_43_ExFls_InstanceIdType](#) InstanceId, [Mem_43_ExFls_AddressType](#) TargetAddress, const [Mem_43_ExFls_DataType](#) *SourceDataPtr, [Mem_43_ExFls_LengthType](#) Length)
Writes to flash memory.

- Std_ReturnType [Mem_43_ExFls_Erase](#) ([Mem_43_ExFls_InstanceIdType](#) InstanceId, [Mem_43_ExFls_AddressType](#) TargetAddress, [Mem_43_ExFls_LengthType](#) Length)
Erase one or more complete flash sectors.
- Std_ReturnType [Mem_43_ExFls_BlankCheck](#) ([Mem_43_ExFls_InstanceIdType](#) InstanceId, [Mem_43_ExFls_AddressType](#) TargetAddress, [Mem_43_ExFls_LengthType](#) Length)
Verify whether a given memory area has been erased but not (yet) programmed.
- Std_ReturnType [Mem_43_ExFls_HwSpecificService](#) ([Mem_43_ExFls_InstanceIdType](#) InstanceId, [Mem_43_ExFls_HwServiceIdType](#) HwServiceId, [Mem_43_ExFls_DataType](#) *DataPtr, [Mem_43_ExFls_LengthType](#) *LengthPtr)
Trigger a hardware specific service.
- void [Mem_43_ExFls_MainFunction](#) (void)
Service to handle the requested jobs and the internal management operations.
- Std_ReturnType [Mem_43_ExFls_IPW_Init](#) (void)
Initialize the hardware resources.
- [Mem_43_ExFls_JobResultType](#) [Mem_43_ExFls_IPW_GetJobResult](#) (uint32 InstanceIndex)
Returns synchronously the result of the last job.
- void [Mem_43_ExFls_IPW_AbortSuspended](#) (uint32 InstanceIndex)
Abort a suspended hardware job to prepare for a new job.
- [Mem_43_ExFls_JobResultType](#) [Mem_43_ExFls_IPW_Read](#) (uint32 InstanceIndex, [Mem_43_ExFls_JobRuntimeInfoType](#) *JobInfo)
IP wrapper read function.
- [Mem_43_ExFls_JobResultType](#) [Mem_43_ExFls_IPW_BlankCheck](#) (uint32 InstanceIndex, [Mem_43_ExFls_JobRuntimeInfoType](#) *JobInfo)
IP wrapper blank check function.
- [Mem_43_ExFls_JobResultType](#) [Mem_43_ExFls_IPW_Write](#) (uint32 InstanceIndex, [Mem_43_ExFls_JobRuntimeInfoType](#) *JobInfo)
IP wrapper write function.
- [Mem_43_ExFls_JobResultType](#) [Mem_43_ExFls_IPW_Erase](#) (uint32 InstanceIndex, [Mem_43_ExFls_JobRuntimeInfoType](#) *JobInfo)
IP wrapper erase function.

6.1.2 Data Structure Documentation

6.1.2.1 struct [Mem_43_ExFls_MemDeviceType](#)

Mem device config type.

Mem device config data structure

Definition at line 186 of file [Mem_43_ExFls_Types.h](#).

Data Fields

	Type	Name	Description
	const Mem_43_ExFls_QspiUnitType *	QspiUnits	QSPI unit configurations (size = MEM_EXFLS_QSPI_UNIT_COUNT)
	const Mem_43_ExFls_MemoryUnitType *	MemUnits	Flash memory unit configurations (size = MEM_EXFLS_MEM_UNIT_COUNT) Each "MemUnit" = a pair of "MemConnection" +
146	S32K3	MEM_EXFLS_Driver	NXP Semiconductors

6.1.2.2 struct Mem_43_ExFls_SectorBatchType

Sector Batch Type.

Sector Batch data structure for group of identical sectors Note: burst sizes equal to normal sizes in case burst disabled

Definition at line 205 of file [Mem_43_ExFls_Types.h](#).

6.1.2.3 struct Mem_43_ExFls_MemInstanceType

Mem Instance Type.

Mem Instance data structure

Definition at line 222 of file [Mem_43_ExFls_Types.h](#).

6.1.2.4 struct Mem_43_ExFls_ConfigType

Mem Configuration Type.

Mem module initialization data structure

Definition at line 236 of file [Mem_43_ExFls_Types.h](#).

6.1.2.5 struct Mem_43_ExFls_JobRuntimeInfoType

Mem job runtime information Type.

This structure contains runtime information the current processing job of each Mem instance.

Definition at line 251 of file [Mem_43_ExFls_Types.h](#).

6.1.3 Macro Definition Documentation

6.1.3.1 MEM_43_EXFLS_MODULE_ID

```
#define MEM_43_EXFLS_MODULE_ID
```

AUTOSAR module identification.

Definition at line 76 of file [Mem_43_ExFls.h](#).

6.1.3.2 MEM_43_EXFLS_INSTANCE_ID

```
#define MEM_43_EXFLS_INSTANCE_ID
```

AUTOSAR module instance identification.

Definition at line 81 of file [Mem_43_ExFls.h](#).

6.1.3.3 MEM_43_EXFLS_E_UNINIT

```
#define MEM_43_EXFLS_E_UNINIT
```

Development error codes (passed to DET)

Definition at line 88 of file [Mem_43_ExFls.h](#).

6.1.3.4 MEM_43_EXFLS_DEINIT_ID

```
#define MEM_43_EXFLS_DEINIT_ID
```

All service IDs (passed to DET)

Definition at line 101 of file [Mem_43_ExFls.h](#).

6.1.4 Types Reference

6.1.4.1 Mem_43_ExFls_AddressType

```
typedef uint32 Mem_43_ExFls_AddressType
```

Mem Address Type.

Physical memory device address type

Definition at line 88 of file [Mem_43_ExFls_Types.h](#).

6.1.4.2 Mem_43_ExFls_InstanceIdType

```
typedef uint32 Mem_43_ExFls_InstanceIdType
```

Mem Instance Id Type.

Mem Instance Id Type

Definition at line 96 of file [Mem_43_ExFls_Types.h](#).

6.1.4.3 Mem_43_ExFls_LengthType

```
typedef uint32 Mem_43_ExFls_LengthType
```

Mem Length Type.

Physical memory device length type

Definition at line 104 of file [Mem_43_ExFls_Types.h](#).

6.1.4.4 Mem_43_ExFls_DataType

```
typedef uint8 Mem_43_ExFls_DataType
```

Mem Data Type.

Read data user buffer type

Definition at line 112 of file [Mem_43_ExFls_Types.h](#).

6.1.4.5 Mem_43_ExFls_CrcType

```
typedef uint16 Mem_43_ExFls_CrcType
```

Mem CRC Type.

CRC computed over config set

Definition at line 119 of file [Mem_43_ExFls_Types.h](#).

6.1.4.6 Mem_43_ExFls_HwServiceIdType

```
typedef uint32 Mem_43_ExFls_HwServiceIdType
```

Mem Hardware Service Id Type.

Hardware specific service request identifier type

Definition at line 127 of file [Mem_43_ExFls_Types.h](#).

6.1.5 Enum Reference

6.1.5.1 Mem_43_ExFls_JobResultType

```
enum Mem_43_ExFls_JobResultType
```

Mem job result type.

Definition at line 138 of file [Mem_43_ExFls_Types.h](#).

6.1.5.2 Mem_43_ExFls_JobType

```
enum Mem_43_ExFls_JobType
```

Type of job currently executed by Mem_43_ExFls_MainFunction.

Definition at line 152 of file [Mem_43_ExFls_Types.h](#).

6.1.6 Function Reference

6.1.6.1 Mem_43_ExFls_Init()

```
void Mem_43_ExFls_Init (
    const Mem_43_ExFls_ConfigType * ConfigPtr )
```

The function initializes Mem_43_ExFls module.

6.1.6.2 Mem_43_ExFls_DeInit()

```
void Mem_43_ExFls_DeInit (
    void )
```

The function de-initializes the Mem_43_ExFls module.

6.1.6.3 Mem_43_ExFls_GetVersionInfo()

```
void Mem_43_ExFls_GetVersionInfo (
    Std_VersionInfoType * VersionInfoPtr )
```

Return the version information of the Mem module.

6.1.6.4 Mem_43_ExFls_GetJobResult()

```
Mem_43_ExFls_JobResultType Mem_43_ExFls_GetJobResult (
    Mem_43_ExFls_InstanceIdType InstanceId )
```

Returns the result of the most recent job.

6.1.6.5 Mem_43_ExFls_Suspend()

```
void Mem_43_ExFls_Suspend (
    Mem_43_ExFls_InstanceIdType InstanceId )
```

Suspends active memory operation using hardware mechanism.

6.1.6.6 Mem_43_ExFls_Resume()

```
void Mem_43_ExFls_Resume (
    Mem_43_ExFls_InstanceIdType InstanceId )
```

Resumes suspended memory operation using hardware mechanism.

6.1.6.7 Mem_43_ExFls_PropagateError()

```
void Mem_43_ExFls_PropagateError (
    Mem_43_ExFls_InstanceIdType InstanceId )
```

Propagates an ECC error to the memory upper layers.

6.1.6.8 Mem_43_ExFls_Read()

```
Std_ReturnType Mem_43_ExFls_Read (
    Mem_43_ExFls_InstanceIdType InstanceId,
    Mem_43_ExFls_AddressType SourceAddress,
    Mem_43_ExFls_DataType * DestinationDataPtr,
    Mem_43_ExFls_LengthType Length )
```

Reads from flash memory.

6.1.6.9 Mem_43_ExFls_Write()

```
Std_ReturnType Mem_43_ExFls_Write (
    Mem_43_ExFls_InstanceIdType InstanceId,
    Mem_43_ExFls_AddressType TargetAddress,
    const Mem_43_ExFls_DataType * SourceDataPtr,
    Mem_43_ExFls_LengthType Length )
```

Writes to flash memory.

6.1.6.10 Mem_43_ExFls_Erase()

```
Std_ReturnType Mem_43_ExFls_Erase (
    Mem_43_ExFls_InstanceIdType InstanceId,
    Mem_43_ExFls_AddressType TargetAddress,
    Mem_43_ExFls_LengthType Length )
```

Erase one or more complete flash sectors.

6.1.6.11 Mem_43_ExFls_BlankCheck()

```
Std_ReturnType Mem_43_ExFls_BlankCheck (
    Mem_43_ExFls_InstanceIdType InstanceId,
    Mem_43_ExFls_AddressType TargetAddress,
    Mem_43_ExFls_LengthType Length )
```

Verify whether a given memory area has been erased but not (yet) programmed.

6.1.6.12 Mem_43_ExFls_HwSpecificService()

```
Std_ReturnType Mem_43_ExFls_HwSpecificService (
    Mem_43_ExFls_InstanceIdType InstanceId,
    Mem_43_ExFls_HwServiceIdType HwServiceId,
    Mem_43_ExFls_DataType * DataPtr,
    Mem_43_ExFls_LengthType * LengthPtr )
```

Trigger a hardware specific service.

6.1.6.13 Mem_43_ExFls_MainFunction()

```
void Mem_43_ExFls_MainFunction (
    void )
```

Service to handle the requested jobs and the internal management operations.

6.1.6.14 Mem_43_ExFls_IPW_Init()

```
Std_ReturnType Mem_43_ExFls_IPW_Init (
    void )
```

Initialize the hardware resources.

6.1.6.15 Mem_43_ExFls_IPW_GetJobResult()

```
Mem_43_ExFls_JobResultType Mem_43_ExFls_IPW_GetJobResult (
    uint32 InstanceIndex )
```

Returns synchronously the result of the last job.

6.1.6.16 Mem_43_ExFls_IPW_AbortSuspended()

```
void Mem_43_ExFls_IPW_AbortSuspended (
    uint32 InstanceIndex )
```

Abort a suspended hardware job to prepare for a new job.

6.1.6.17 Mem_43_ExFls_IPW_Read()

```
Mem_43_ExFls_JobResultType Mem_43_ExFls_IPW_Read (
    uint32 InstanceIndex,
    Mem_43_ExFls_JobRuntimeInfoType * JobInfo )
```

IP wrapper read function.

6.1.6.18 Mem_43_ExFls_IPW_BlankCheck()

```
Mem_43_ExFls_JobResultType Mem_43_ExFls_IPW_BlankCheck (
    uint32 InstanceIndex,
    Mem_43_ExFls_JobRuntimeInfoType * JobInfo )
```

IP wrapper blank check function.

6.1.6.19 Mem_43_ExFls_IPW_Write()

```
Mem_43_ExFls_JobResultType Mem_43_ExFls_IPW_Write (
    uint32 InstanceIndex,
    Mem_43_ExFls_JobRuntimeInfoType * JobInfo )
```

IP wrapper write function.

6.1.6.20 Mem_43_ExFls_IPW_Erase()

```
Mem_43_ExFls_JobResultType Mem_43_ExFls_IPW_Erase (
    uint32 InstanceIndex,
    Mem_43_ExFls_JobRuntimeInfoType * JobInfo )
```

IP wrapper erase function.

6.2 QSPI IPV Driver

6.2.1 Detailed Description

Data Structures

- struct [Qspi_Ip_HyperFlashConfigType](#)
Hyperflash configuration structure. [More...](#)
- struct [Qspi_Ip_DllSettingsType](#)
DLL configuration structure. [More...](#)
- 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`
- `#define QSPI_IP_PACK_LUT_REG(ops0, ops1)`

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_HyperflashParamSectorMapType](#)
Parameter sector map.
- enum [Qspi_Ip_HyperflashDrvStrengthType](#)
Drive strength.
- enum [Qspi_Ip_HyperflashReadLatencyType](#)
Read latency.
- enum [Qspi_Ip_HyperflashAsoEntryCommandsType](#)
- enum [Qspi_Ip_HyperflashSectorProtectionType](#)
Sector protection type.
- enum [Qspi_Ip_StatusType](#)
qspi return codes
- enum [Qspi_Ip_InstanceType](#)
qspi hardware instance
- 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.

- enum [Qspi_Ip_DllModeType](#)

DLL configuration modes.

- enum [Qspi_Ip_Sfp_AccessControlType](#)
- enum [Qspi_Ip_SfpEalType](#)
- enum [Qspi_Ip_SfpAcpType](#)
- enum [Qspi_Ip_LastCommandType](#)

Last command that was executed by the device flash.

- enum [Qspi_Ip_FlashMemoryType](#)

Parameter memory type.

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_MemoryConnectionType](#) *pConnect)
Initializes the serial flash memory configuration from SFDP table.

QuadSPI Driver

- void [Qspi_Ip_SetLut](#) (uint32 instance, uint8 LutRegister, [Qspi_Ip_InstrOpType](#) operation0, [Qspi_Ip_InstrOpType](#) operation1)
Configures LUT commands.
- void [Qspi_Ip_WriteLuts_Privileged](#) (uint32 Instance, uint8 StartLutRegister, const uint32 *Data, uint8 Size)
Configures LUT commands.
- void [Qspi_Ip_SetAhbSeqId_Privileged](#) (uint32 instance, uint8 seqID)
Sets sequence ID for AHB operations.
- uint32 [Qspi_Ip_GetBaseAdress](#) (uint32 instance, [Qspi_Ip_ConnectionType](#) connectionType)
Returns the physical base address of a flash device.
- [Qspi_Ip_StatusType Qspi_Ip_IpCommand](#) (uint32 instance, uint8 SeqId, uint32 addr)
Launches a simple IP command.
- [Qspi_Ip_StatusType Qspi_Ip_IpRead](#) (uint32 instance, uint8 SeqId, uint32 addr, uint8 *dataRead, const uint8 *dataCmp, uint32 size)
Launches an IP read command.
- [Qspi_Ip_StatusType Qspi_Ip_IpWrite](#) (uint32 instance, uint8 SeqId, uint32 addr, const uint8 *data, uint32 size)
Launches an IP write command.
- #define **MEM_43_EXFLS_START_SEC_CODE**
- #define **MEM_43_EXFLS_STOP_SEC_CODE**

6.2.2 Data Structure Documentation

6.2.2.1 struct Qspi_Ip_HyperFlashConfigType

Hyperflash configuration structure.

This structure is used to provide configuration parameters for HyperFlash at initialization time.

Definition at line 216 of file [Qspi_Ip_HyperflashTypes.h](#).

Data Fields

Type	Name	Description
Qspi_Ip_HyperflashDrvStrengthType	outputDriverStrength	Output driver level of the device
boolean	RWDSLowOnDualError	Specifies if RWDS will stall upon Dual Error Detect
boolean	secureRegionUnlocked	If true, the secure silicon region will be locked
Qspi_Ip_HyperflashReadLatencyType	readLatency	Read latency
Qspi_Ip_HyperflashParamSectorMapType	paramSectorMap	Parameter sector mapping
uint32	deviceIdWordAddress	The word address of the device Id in ASO

6.2.2.2 struct Qspi_Ip_DllSettingsType

DLL configuration structure.

This structure contains initialization settings for DLL and slave delay chain

Definition at line 327 of file [Qspi_Ip_Types.h](#).

Data Fields

Type	Name	Description
Qspi_Ip_DllModeType	dllMode	Mode in which DLL is used
boolean	freqEnable	Selects delay-chain for high frequency of operation
uint8	referenceCounter	Select the "n+1" interval of DLL phase detection and reference delay updating interval
uint8	resolution	Minimum resolution for DLL phase detector

Data Fields

Type	Name	Description
uint8	coarseDelay	Coarse delay DLL slave delay chain
uint8	fineDelay	Fine delay DLL slave delay chain
uint8	tapSelect	Selects the Nth tap provided by the slave delay-chain

6.2.2.3 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 344 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

6.2.2.4 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 464 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
uint8	csHoldTime	CS hold time, expressed in serial clock cycles

Data Fields

Type	Name	Description
uint8	csSetupTime	CS setup time, expressed in serial clock cycles
uint8	columnAddr	Width of the column address, 0 if not used
boolean	wordAddressable	True if serial flash is word addressable
Qspi_Ip_ReadModeType	readModeA	Read mode for incoming data from serial flash A
Qspi_Ip_SampleDelayType	sampleDelay	Delay (in clock cycles) used for sampling Rx data
Qspi_Ip_SamplePhaseType	samplePhase	Clock phase used for sampling Rx data
Qspi_Ip_DllSettingsType	dllSettingsA	DLL settings for side A
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
boolean	byteSwap	Enable byte swap in octal DDR mode
Qspi_Ip_SfpConfigType	SfpCfg	
Qspi_Ip_ControllerAhbConfigType	ahbConfig	AHB buffers configuration

6.2.2.5 struct [Qspi_Ip_StatusConfigType](#)

Status register configuration structure.

This structure contains information about the status registers of the external flash

Definition at line 506 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

Data Fields

Type	Name	Description
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

6.2.2.6 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 528 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

6.2.2.7 struct Qspi_Ip_EraseConfigType

Erase capabilities configuration structure.

This structure contains information about the erase capabilities of the external flash

Definition at line 540 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

6.2.2.8 struct Qspi_Ip_ReadIdConfigType

Read Id capabilities configuration structure.

This structure contains information about the read manufacturer/device ID command

Definition at line 552 of file [Qspi_Ip_Types.h](#).

Data Fields

Type	Name	Description
uint16	readIdLut	Read Id command
uint8	readIdSize	Size of data returned by Read Id command
uint8	readIdExpected[10U]	Device ID configured value (Memory density Memory type Manufacturer ID)

6.2.2.9 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 565 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

6.2.2.10 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 579 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

6.2.2.11 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 607 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

6.2.2.12 struct Qspi_Ip_InitOperationType

Initialization operation.

This structure describes one initialization operation.

Definition at line 619 of file [Qspi_Ip_Types.h](#).

Data Fields

Type	Name	Description
Qspi_Ip_OpType	opType	Operation type
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

Data Fields

Type	Name	Description
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

6.2.2.13 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 640 of file [Qspi_Ip_Types.h](#).

Data Fields

Type	Name	Description
uint8	opCount	Number of operations
Qspi_Ip_InitOperationType *	operations	List of operations

6.2.2.14 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 663 of file [Qspi_Ip_Types.h](#).

Data Fields

Type	Name	Description
Qspi_Ip_FlashMemoryType	memType	Mmemory device type

Data Fields

Type	Name	Description
const Qspi_Ip_HyperFlashConfigType *	hfConfig	Hyperflash configuration, NULL if not used
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

6.2.2.15 struct Qspi_Ip_MemoryConnectionType

Flash-controller connections configuration structure.

This structure specifies the connections of each flash device to QSPI controllers at initialization time.

Definition at line 695 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

6.2.3 Macro Definition Documentation

6.2.3.1 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 100 of file [Qspi_Ip.h](#).

6.2.3.2 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 102 of file [Qspi_Ip.h](#).

6.2.3.3 QSPI_IP_ERASE_TYPES

```
#define QSPI_IP_ERASE_TYPES
```

Number of erase types that can be supported by a flash device

Definition at line 131 of file [Qspi_Ip_Types.h](#).

6.2.3.4 QSPI_IP_AHB_BUFFERS

```
#define QSPI_IP_AHB_BUFFERS
```

Number of AHB buffers in the device.

Definition at line 134 of file [Qspi_Ip_Types.h](#).

6.2.3.5 QSPI_IP_LUT_INVALID

```
#define QSPI_IP_LUT_INVALID
```

Invalid index in virtual LUT, used for unsupported features

Definition at line 137 of file [Qspi_Ip_Types.h](#).

6.2.3.6 QSPI_IP_LUT_SEQ_END

```
#define QSPI_IP_LUT_SEQ_END
```

End operation for a LUT sequence

Definition at line 140 of file [Qspi_Ip_Types.h](#).

6.2.3.7 QSPI_IP_PACK_LUT_REG

```
#define QSPI_IP_PACK_LUT_REG(  
    ops0,  
    ops1 )
```

Pack the two operations into a LUT register (each operation is a pair of instruction-operand)

Definition at line 143 of file [Qspi_Ip_Types.h](#).

6.2.4 Types Reference

6.2.4.1 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 240 of file [Qspi_Ip__Types.h](#).

6.2.4.2 Qspi_Ip__InitCalloutPtrType

```
typedef Qspi_Ip__StatusType(* Qspi_Ip__InitCalloutPtrType) (uint32 instance)
```

Init callout pointer type.

Definition at line 303 of file [Qspi_Ip__Types.h](#).

6.2.4.3 Qspi_Ip__ResetCalloutPtrType

```
typedef Qspi_Ip__StatusType(* Qspi_Ip__ResetCalloutPtrType) (uint32 instance)
```

Reset callout pointer type.

Definition at line 307 of file [Qspi_Ip__Types.h](#).

6.2.4.4 Qspi_Ip__ErrorCheckCalloutPtrType

```
typedef Qspi_Ip__StatusType(* Qspi_Ip__ErrorCheckCalloutPtrType) (uint32 instance)
```

Error Check callout pointer type.

Definition at line 311 of file [Qspi_Ip__Types.h](#).

6.2.4.5 Qspi_Ip_EccCheckCalloutPtrType

```
typedef Qspi_Ip_StatusType (* Qspi_Ip_EccCheckCalloutPtrType) (uint32 instance, uint32 startAddress, uint32
dataLength)
```

Ecc Check callout pointer type.

Definition at line 315 of file [Qspi_Ip_Types.h](#).

6.2.5 Enum Reference

6.2.5.1 Qspi_Ip_HyperflashParamSectorMapType

```
enum Qspi_Ip_HyperflashParamSectorMapType
```

Parameter sector map.

This structure is used to configure how the Parameter-Sectors are used and how they are mapped into the address map.

Definition at line 130 of file [Qspi_Ip_HyperflashTypes.h](#).

6.2.5.2 Qspi_Ip_HyperflashDrvStrengthType

```
enum Qspi_Ip_HyperflashDrvStrengthType
```

Drive strength.

Hyperflash driver strength settings.

Enumerator

QSPI_IP_HF_DRV_STRENGTH_000	Typical Impedance for 1.8V: 27, Typical Impedance 3V: 20
QSPI_IP_HF_DRV_STRENGTH_001	Typical Impedance for 1.8V: 117, Typical Impedance 3V: 71
QSPI_IP_HF_DRV_STRENGTH_002	Typical Impedance for 1.8V: 68, Typical Impedance 3V: 40
QSPI_IP_HF_DRV_STRENGTH_003	Typical Impedance for 1.8V: 45, Typical Impedance 3V: 27
QSPI_IP_HF_DRV_STRENGTH_004	Typical Impedance for 1.8V: 34, Typical Impedance 3V: 20
QSPI_IP_HF_DRV_STRENGTH_005	Typical Impedance for 1.8V: 27, Typical Impedance 3V: 16
QSPI_IP_HF_DRV_STRENGTH_006	Typical Impedance for 1.8V: 24, Typical Impedance 3V: 14
QSPI_IP_HF_DRV_STRENGTH_007	Typical Impedance for 1.8V: 20, Typical Impedance 3V: 12

Definition at line 144 of file [Qspi_Ip_HyperflashTypes.h](#).

6.2.5.3 Qspi_Ip_HyperflashReadLatencyType

enum [Qspi_Ip_HyperflashReadLatencyType](#)

Read latency.

Enumerator

QSPI_IP_HF_READ_LATENCY_5_CLOCKS	Read latency 5 clocks
QSPI_IP_HF_READ_LATENCY_6_CLOCKS	Read latency 6 clocks
QSPI_IP_HF_READ_LATENCY_7_CLOCKS	Read latency 7 clocks
QSPI_IP_HF_READ_LATENCY_8_CLOCKS	Read latency 8 clocks
QSPI_IP_HF_READ_LATENCY_9_CLOCKS	Read latency 9 clocks
QSPI_IP_HF_READ_LATENCY_10_CLOCKS	Read latency 10 clocks
QSPI_IP_HF_READ_LATENCY_11_CLOCKS	Read latency 11 clocks
QSPI_IP_HF_READ_LATENCY_12_CLOCKS	Read latency 12 clocks
QSPI_IP_HF_READ_LATENCY_13_CLOCKS	Read latency 13 clocks
QSPI_IP_HF_READ_LATENCY_14_CLOCKS	Read latency 14 clocks
QSPI_IP_HF_READ_LATENCY_15_CLOCKS	Read latency 15 clocks
QSPI_IP_HF_READ_LATENCY_16_CLOCKS	Read latency 16 clocks

Definition at line 161 of file [Qspi_Ip_HyperflashTypes.h](#).

6.2.5.4 Qspi_Ip_HyperflashAsoEntryCommandsType

enum [Qspi_Ip_HyperflashAsoEntryCommandsType](#)

Enumerator

QSPI_IP_HF_PASSWORD_ASO_ENTRY	Password ASO Entry command
QSPI_IP_HF_PPB_ASO_ENTRY	PPB ASO Entry command
QSPI_IP_HF_PPB_LOCK_ASO_ENTRY	PPB Lock ASO Entry command
QSPI_IP_HF_DYB_ASO_ENTRY	DYB ASO Entry command
QSPI_IP_HF_ECC_ASO_ENTRY	ECC ASO Entry command
QSPI_IP_HF_SSR_ASO_ENTRY	Secure Silicon Region command

Enumerator

QSPI_IP_HF_CRC_ASO_ENTRY	CRC ASO Entry command
QSPI_IP_HF_ASPR_ASO_ENTRY	ASP Configuration Register ASO entry command
QSPI_IP_HF_FLASH_MEMORY_ARRAY	No ASO entry

Definition at line 179 of file [Qspi_Ip_HyperflashTypes.h](#).

6.2.5.5 Qspi_Ip_HyperflashSectorProtectionType

enum [Qspi_Ip_HyperflashSectorProtectionType](#)

Sector protection type.

Definition at line 197 of file [Qspi_Ip_HyperflashTypes.h](#).

6.2.5.6 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_VERIFY	Error - selected memory area doesn't contain desired value
STATUS_QSPI_IP_ERROR_BLANK_CHECK	Error - selected memory area isn't in erased state

Definition at line 152 of file [Qspi_Ip_Types.h](#).

6.2.5.7 Qspi_Ip_InstanceType

enum [Qspi_Ip_InstanceType](#)

qspi hardware instance

Enumerator

QSPI_IP_INSTANCE↵ _0	QuadSPI hardware instance 0
QSPI_IP_INSTANCE↵ _1	QuadSPI hardware instance 1

Definition at line 165 of file [Qspi_Ip_Types.h](#).

6.2.5.8 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 174 of file [Qspi_Ip_Types.h](#).

6.2.5.9 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
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 185 of file [Qspi_Ip_Types.h](#).

6.2.5.10 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
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 196 of file [Qspi_Ip_Types.h](#).

6.2.5.11 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 223 of file [Qspi_Ip_Types.h](#).

6.2.5.12 Qspi_Ip_ReadModeType

enum [Qspi_Ip_ReadModeType](#)

Read mode.

Enumerator

QSPI_IP_READ_MODE_LOOPBACK	Use loopback clock from PAD as strobe signal
QSPI_IP_READ_MODE_EXTERNAL_DQS	Use external strobe signal

Definition at line 244 of file [Qspi_Ip_Types.h](#).

6.2.5.13 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 261 of file [Qspi_Ip_Types.h](#).

6.2.5.14 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 270 of file [Qspi_Ip_Types.h](#).

6.2.5.15 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 278 of file [Qspi_Ip_Types.h](#).

6.2.5.16 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 286 of file [Qspi_Ip_Types.h](#).

6.2.5.17 Qspi_Ip_DllModeType

enum [Qspi_Ip_DllModeType](#)

DLL configuration modes.

Enumerator

QSPI_IP_DLL_BYPASSED	DLL bypass mode
QSPI_IP_DLL_MANUAL_UPDATE	DLL manual update mode
QSPI_IP_DLL_AUTO_UPDATE	DLL auto update mode

Definition at line 294 of file [Qspi_Ip_Types.h](#).

6.2.5.18 Qspi_Ip_Sfp_AccessControlType

enum [Qspi_Ip_Sfp_AccessControlType](#)

Access control type

Definition at line 373 of file [Qspi_Ip_Types.h](#).

6.2.5.19 Qspi_Ip_SfpEalType

enum [Qspi_Ip_SfpEalType](#)

FRADn_WORD3 :: Exclusive access Lock [EAL]

Definition at line [412](#) of file [Qspi_Ip_Types.h](#).

6.2.5.20 Qspi_Ip_SfpAcpType

enum [Qspi_Ip_SfpAcpType](#)

Access Control Policy for write operations

Definition at line [420](#) of file [Qspi_Ip_Types.h](#).

6.2.5.21 Qspi_Ip_LastCommandType

enum [Qspi_Ip_LastCommandType](#)

Last command that was executed by the device flash.

Definition at line [589](#) of file [Qspi_Ip_Types.h](#).

6.2.5.22 Qspi_Ip_FlashMemoryType

enum [Qspi_Ip_FlashMemoryType](#)

Parameter memory type.

Enumerator

QSPI_IP_HYPER_FLASH	Hyperbus devices
QSPI_IP_SERIAL_FLASH	Traditional xSPI devices

Definition at line [650](#) of file [Qspi_Ip_Types.h](#).

6.2.6 Function Reference

6.2.6.1 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

<i>instance</i>	External flash instance number
<i>pConfig</i>	Pointer to the driver configuration structure.
<i>pConnect</i>	Pointer to the flash device connection structure.

Returns

Error or success status returned by API

6.2.6.2 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

<i>instance</i>	External flash instance number
-----------------	--------------------------------

Returns

void

6.2.6.3 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

<i>instance</i>	External flash instance number
<i>address</i>	Address of sector to be erased
<i>size</i>	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

6.2.6.4 Qspi_Ip_EraseChip()

```
Qspi_Ip_StatusType Qspi_Ip_EraseChip (
uint32 instance )
```

Erase the entire serial flash.

Parameters

<i>instance</i>	External flash instance number
-----------------	--------------------------------

Returns

Error or success status returned by API

6.2.6.5 Qspi_Ip_GetMemoryStatus()

```
Qspi_Ip_StatusType Qspi_Ip_GetMemoryStatus (
uint32 instance )
```

Check the status of the flash device.

Parameters

<i>instance</i>	External flash instance number
-----------------	--------------------------------

Returns

Error or success status returned by API

6.2.6.6 Qspi_Ip_SetProtection()

```
Qspi_Ip_StatusType Qspi_Ip_SetProtection (
    uint32 instance,
    uint8 value )
```

Sets the protection bits to the requested value.

Parameters

<i>instance</i>	External flash instance number
<i>value</i>	New value for the protection bits

Returns

Error or success status returned by API

6.2.6.7 Qspi_Ip_GetProtection()

```
Qspi_Ip_StatusType Qspi_Ip_GetProtection (
    uint32 instance,
    uint8 * value )
```

Returns the current value of the protection bits.

Parameters

<i>instance</i>	External flash instance number
<i>value</i>	Current value of the protection bits

Returns

Error or success status returned by API

6.2.6.8 Qspi_Ip_Reset()

```
Qspi_Ip_StatusType Qspi_Ip_Reset (
    uint32 instance )
```

Resets the flash device.

Parameters

<i>instance</i>	External flash instance number
-----------------	--------------------------------

Returns

Error or success status returned by API

6.2.6.9 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

<i>instance</i>	External flash instance number
-----------------	--------------------------------

Returns

Error or success status returned by API

6.2.6.10 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

<i>instance</i>	External flash instance number
-----------------	--------------------------------

Returns

Error or success status returned by API

6.2.6.11 Qspi_Ip_ProgramSuspend()

```
Qspi_Ip_StatusType Qspi_Ip_ProgramSuspend (
    uint32 instance )
```

Suspends a program operation.

Parameters

<i>instance</i>	External flash instance number
-----------------	--------------------------------

Returns

Error or success status returned by API

6.2.6.12 Qspi_Ip_ProgramResume()

```
Qspi_Ip_StatusType Qspi_Ip_ProgramResume (
    uint32 instance )
```

Resumes a program operation.

Parameters

<i>instance</i>	External flash instance number
-----------------	--------------------------------

Returns

Error or success status returned by API

6.2.6.13 Qspi_Ip_EraseSuspend()

```
Qspi_Ip_StatusType Qspi_Ip_EraseSuspend (
    uint32 instance )
```

Suspends an erase operation.

Parameters

<i>instance</i>	External flash instance number
-----------------	--------------------------------

Returns

Error or success status returned by API

6.2.6.14 Qspi_Ip_EraseResume()

```
Qspi_Ip_StatusType Qspi_Ip_EraseResume (
    uint32 instance )
```

Resumes an erase operation.

Parameters

<i>instance</i>	External flash instance number
-----------------	--------------------------------

Returns

Error or success status returned by API

6.2.6.15 Qspi_Ip_Read()

```
Qspi_Ip_StatusType Qspi_Ip_Read (
    uint32 instance,
    uint32 address,
    uint8 * data,
    uint32 size )
```

Read data from serial flash.

Parameters

<i>instance</i>	External flash instance number
<i>address</i>	Start address for read operation
<i>data</i>	Buffer where to store read data
<i>size</i>	Size of data buffer

Returns

Error or success status returned by API

6.2.6.16 Qspi_Ip_ReadId()

```
Qspi_Ip_StatusType Qspi_Ip_ReadId (
    uint32 instance,
    uint8 * data )
```

Read manufacturer ID/device ID from serial flash.

Parameters

<i>instance</i>	External flash instance number
<i>data</i>	Buffer where to store read data. Buffer size must match ReadId initialization settings.

Returns

Error or success status returned by API

6.2.6.17 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

<i>instance</i>	External flash instance number
<i>address</i>	Start address of area to be verified
<i>data</i>	Data to be verified
<i>size</i>	Size of area to be verified

Returns

Error or success status returned by API

6.2.6.18 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

<i>instance</i>	External flash instance number
<i>address</i>	Start address of area to be verified
<i>size</i>	Size of area to be verified

Returns

Error or success status returned by API

6.2.6.19 Qspi_Ip_Program()

```
Qspi_Ip_StatusType Qspi_Ip_Program (
    uint32 instance,
    uint32 address,
    const uint8 * data,
    uint32 size )
```

Writes data in serial flash.

Writes data in serial flash memory then exits (Async mode) The status of the flash memory must be verified by calling asynchronously the Qspi_Ip_GetMemoryStatus function until it is not busy, meaning that the write operation is complete. The maximum supported size is equal to the Qspi hardware TxBuffer size.

Parameters

<i>instance</i>	External flash instance number
<i>address</i>	Start address of area to be programmed
<i>data</i>	Data to be programmed in flash
<i>size</i>	Size of data buffer

Returns

Error or success status returned by API

6.2.6.20 Qspi_Ip_RunCommand()

```
Qspi_Ip_StatusType Qspi_Ip_RunCommand (
    uint32 instance,
    uint16 lut,
    uint32 addr )
```

Launches a simple command for the serial flash.

Parameters

<i>instance</i>	External flash instance number
<i>lut</i>	Index of command in virtual LUT
<i>addr</i>	Address used in the command, or base address of the target serial flash

Returns

Error or success status returned by API

6.2.6.21 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

Parameters

<i>instance</i>	External flash instance number
<i>lut</i>	Index of command in virtual LUT
<i>addr</i>	Start address for read operation in serial flash
<i>dataRead</i>	Buffer where to store read data
<i>dataCmp</i>	Buffer to be compared to read data
<i>size</i>	Size of data buffer

Returns

Error or success status returned by API

6.2.6.22 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

<i>instance</i>	External flash instance number
<i>lut</i>	Index of command in virtual LUT
<i>addr</i>	Start address for write operation in serial flash
<i>data</i>	Data to be programmed in flash
<i>size</i>	Size of data buffer

Returns

Error or success status returned by API

6.2.6.23 Qspi_Ip_AhbReadEnable()

```
Qspi_Ip_StatusType Qspi_Ip_AhbReadEnable (
    uint32 instance )
```

Sets up AHB reads to the serial flash.

Parameters

<i>instance</i>	External flash instance number
-----------------	--------------------------------

Returns

Error or success status returned by API

6.2.6.24 Qspi_Ip_ControllerGetStatus()

```
Qspi_Ip_StatusType Qspi_Ip_ControllerGetStatus (
    uint32 instance )
```

Check the status of the QSPI controller.

Parameters

<i>instance</i>	QSPI peripheral instance number
-----------------	---------------------------------

Returns

Error or success status returned by API

6.2.6.25 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

<i>instance</i>	QSPI peripheral instance number
<i>userConfigPtr</i>	Pointer to the qspi configuration structure.

Returns

Error or success status returned by API

6.2.6.26 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

<i>instance</i>	QSPI peripheral instance number
-----------------	---------------------------------

Returns

Error or success status returned by API

6.2.6.27 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

<i>pConfig</i>	Pointer to the driver configuration structure.
<i>pConnect</i>	Pointer to the flash device connection structure.

Returns

Error or success status returned by API

6.2.6.28 Qspi_Ip_SetLut()

```
void Qspi_Ip_SetLut (
    uint32 instance,
    uint8 LutRegister,
    Qspi_Ip_InstrOpType operation0,
    Qspi_Ip_InstrOpType operation1 )
```

Configures LUT commands.

This function configures a pair of LUT commands in the specified LUT register. LUT sequences start at index multiple of 4 and can have up to 8 commands

Parameters

<i>instance</i>	QuadSPI peripheral instance number
<i>LutRegister</i>	Index in physical LUT array
<i>operation0</i>	First operation
<i>operation1</i>	Second operation Implements Qspi_Ip_SetLut_Activity

6.2.6.29 Qspi_Ip_WriteLuts_Privileged()

```
void Qspi_Ip_WriteLuts_Privileged (
    uint32 Instance,
    uint8 StartLutRegister,
    const uint32 * Data,
    uint8 Size )
```

Configures LUT commands.

This function configures pairs of LUT commands from the specified LUT register.

Parameters

<i>Instance</i>	QuadSPI peripheral instance number
<i>StartLutRegister</i>	Start index in physical LUT array
<i>Data</i>	Data to be written in LUT register
<i>Size</i>	Size of data buffer

6.2.6.30 Qspi_Ip_SetAhbSeqId_Privileged()

```
void Qspi_Ip_SetAhbSeqId_Privileged (
```

```
uint32 instance,
uint8 seqID )
```

Sets sequence ID for AHB operations.

Parameters

<i>instance</i>	QuadSPI peripheral instance number
<i>seqID</i>	Sequence ID in LUT for read operation Implements Qspi_Ip_SetAhbSeqId_Activity

6.2.6.31 Qspi_Ip_GetBaseAddress()

```
uint32 Qspi_Ip_GetBaseAddress (
    uint32 instance,
    Qspi_Ip_ConnectionType connectionType )
```

Returns the physical base address of a flash device.

This function returns the physical base address of a flash device, depending on the QSPI connection. The controller must be initialized prior to calling this function.

Parameters

<i>instance</i>	QuadSPI peripheral instance number
<i>connectionType</i>	Connection of the flash device to QSPI

6.2.6.32 Qspi_Ip_IpCommand()

```
Qspi_Ip_StatusType Qspi_Ip_IpCommand (
    uint32 instance,
    uint8 SeqId,
    uint32 addr )
```

Launches a simple IP command.

Parameters

<i>instance</i>	QuadSPI peripheral instance number
<i>SeqId</i>	Sequence ID where the command is to be found
<i>addr</i>	Address of the target serial flash

Returns

Error or success status returned by API

6.2.6.33 Qspi_Ip_IpRead()

```
Qspi_Ip_StatusType Qspi_Ip_IpRead (
    uint32 instance,
    uint8 SeqId,
    uint32 addr,
    uint8 * dataRead,
    const uint8 * dataCmp,
    uint32 size )
```

Launches an IP read command.

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

<i>instance</i>	QuadSPI peripheral instance number
<i>SeqId</i>	Sequence ID where the command is to be found
<i>addr</i>	Start address for read operation in serial flash
<i>dataRead</i>	Buffer where to store read data
<i>dataCmp</i>	Buffer to be compared to read data
<i>size</i>	Size of data buffer

Returns

Error or success status returned by API

6.2.6.34 Qspi_Ip_IpWrite()

```
Qspi_Ip_StatusType Qspi_Ip_IpWrite (
    uint32 instance,
```

Module Documentation

```
uint8 SeqId,  
uint32 addr,  
const uint8 * data,  
uint32 size )
```

Launches an IP write command.

Parameters

<i>instance</i>	QuadSPI peripheral instance number
<i>SeqId</i>	Sequence ID where the command is to be found
<i>addr</i>	Start address for write operation in serial flash
<i>data</i>	Data to be programmed in flash
<i>size</i>	Size of data buffer

Returns

Error or success status returned by API

How to Reach Us:

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and Vision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2023 NXP B.V.

