MCAL Configuration Verification Manual for Spi

32-bit TriCore™ AURIX™ TC3xx microcontroller family

About this document

Scope and purpose

This Configuration Data Reference document is applicable to all TC3xx devices in the TriCore™ AURIX™ family of 32-bit microcontrollers.

The purpose of this document is to facilitate the integrator to verify the generated code based on the input configuration parameters. This document describes details of structures, defines, macros and variables generated from the configuration parameters.

Intended audience

This document is intended for integrators who need to understand the logic of the generated configuration code of AURIX™ AUTOSAR MCAL.

Reference documents

This document should be read in conjunction with the following documents:

- AURIX™ TC3xx MCAL User Manual Spi

Table of contents

1 Spi driver

This chapter describes the details of the configuration data generated from the SPI driver.

1.1 File: Spi_Cfg.h

The generated header file contains all pre-compile configuration parameters. Pre-compile time configuration allows decoupling of the static configuration from implementation. The file is generated in the 'inc' folder.

1.1.1 Macro: SPI_AR_RELEASE_MAJOR_VERSION

Table 1 SPI_AR_RELEASE_MAJOR_VERSION

Name	SPI_AR_RELEASE_MAJOR_VERSION	
Description	Major version number of AUTOSAR release on which the Spi implementation is based on.	
Verification method	The macro is generated with the value present in 'CommonPublishedInformation/ArMajorVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Spi_Cfg.h file	`#define SPI_AR_RELEASE_MAJOR_VERSION (4U)`

1.1.2 Macro: SPI_AR_RELEASE_MINOR_VERSION

Table 2 SPI_AR_RELEASE_MINOR_VERSION

Name	SPI_AR_RELEASE_MINOR_VERSION	
Description	Minor version number of AUTOSAR release on which the Spi implementation is based on.	
Verification method	The macro is generated with the value present in 'CommonPublishedInformation/ArMinorVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Spi_Cfg.h file	`#define SPI_AR_RELEASE_MINOR_VERSION (2U)`

1.1.3 Macro: SPI_AR_RELEASE_REVISION_VERSION

Table 3 SPI_AR_RELEASE_REVISION_VERSION

Name	SPI_AR_RELEASE_REVISION_VERSION

Description	Revision version number of AUTOSAR release on which the Spi implementation is based on.	
Verification method	The macro is generated with the value present in 'CommonPublishedInformation/ArPatchVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Spi_Cfg.h file	`#define SPI_AR_RELEASE_REVISION_VERSION (2U)`

1.1.4 Macro: SPI_SW_MAJOR_VERSION

Table 4 SPI_SW_MAJOR_VERSION

Name	SPI_SW_MAJOR_VERSION	
Description	Major version number of the Spi module.	
Verification method	The macro is generated with the value present in 'CommonPublishedInformation/SwMajorVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Spi_Cfg.h file with SwMajorVersion 10	`#define SPI_SW_MAJOR_VERSION (10U)`

1.1.5 Macro: SPI_SW_MINOR_VERSION

Table 5 SPI_SW_MINOR_VERSION

Name	SPI_SW_MINOR_VERSION	
Description	Minor version number of the Spi module.	
Verification method	The macro is generated with the value present in 'CommonPublishedInformation/SwMinorVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Spi_Cfg.h file with SwMinorVersion 10	`#define SPI_SW_MINOR_VERSION (10U)`

1.1.6 Macro: SPI_SW_PATCH_VERSION

Table 6 SPI_SW_PATCH_VERSION

Name	SPI_SW_PATCH_VERSION
Description	Patch level version number of the Spi module.

Verification method	The macro is generated with the value present in 'CommonPublishedInformation/SwPatchVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Spi_Cfg.h file with SwPatchVersion 0	`#define SPI_SW_PATCH_VERSION   (0U)`

1.1.7 Macro: SPI_SAFETY_ENABLE

Table 7 SPI_SAFETY_ENABLE

Name	SPI_SAFETY_ENABLE	
Description	Enables/disables safety features	
Verification method	The macro is generated as STD_ON if SpiSafetyCheckEnable configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	SpiSafetyCheckEnable = True	`#define SPI_SAFETY_ENABLE (STD_ON)`
	SpiSafetyCheckEnable = False	`#define SPI_SAFETY_ENABLE (STD_OFF)`

1.1.8 Macro: SPI_DEM_REPORT_DISABLED

Table 8 SPI_DEM_REPORT_DISABLED

Name	SPI_DEM_REPORT_DISABLED	
Description	Disables the Production Error reporting. *Note: The macro is not user configurable.*	
Verification method	The macro is generated always with value '0'.	
Example(s)	**Action**	**Generated output**
	Generate 'Spi_Cfg.h'	`#define SPI_DEM_REPORT_DISABLED (0)`

1.1.9 Macro: SPI_DEM_REPORT_ENABLED

Table 9 SPI_DEM_REPORT_ENABLED

Name	SPI_DEM_REPORT_ENABLED
Description	Enables the Production Error reporting. *Note: The macro is not user configurable.*
Verification method	The macro is generated always with value '1'.

Example(s)	**Action**	**Generated output**
	Generate 'Spi_Cfg.h'	`#define SPI_DEM_REPORT_ENABLED (1)`

1.1.10 Macro: SPI_HW_ERROR_DEM_REPORT

Table 10 SPI_HW_ERROR_DEM_REPORT

Name	SPI_HW_ERROR_DEM_REPORT	
Description	Enables/disables the reporting of Production Error.	
Verification method	The macro is generated as SPI_DEM_REPORT_ENABLED if SpiDemEventParameterRefs/ SPI_E_HARDWARE_ERROR is configured else the macro is generated as SPI_DEM_REPORT_DISABLED.	
Example(s)	**Action**	**Generated output**
	SpiDemEventParameterRefs/ SPI_E_HARDWARE_ERROR is configured	`#define SPI_HW_ERROR_DEM_REPORT (SPI_DEM_REPORT_ENABLED)`
	SpiDemEventParameterRefs/ SPI_E_HARDWARE_ERROR is not configured	`#define SPI_HW_ERROR_DEM_REPORT (SPI_DEM_REPORT_DISABLED)`

1.1.11 Macro: SPI_E_HARDWARE_ERROR

Table 11 SPI_E_HARDWARE_ERROR

Name	SPI_E_HARDWARE_ERROR	
Description	DEM Event information	
Verification method	The macro is generated only when SpiDemEventParameterRefs/ SPI_E_HARDWARE_ERROR are configured else the macro is not generated.	
Example(s)	**Action**	**Generated output**
	SpiDemEventParameterRefs/ SPI_E_HARDWARE_ERROR is configured with valid reference "HardwareError"	`#define SPI_E_HARDWARE_ERROR (DemConf_DemEventParameter_HardwareError)`
	SpiDemEventParameterRefs/ SPI_E_HARDWARE_ERROR is not configured	`The macro is not generated.`

1.1.12 Macro: SPI_MCAL_SUPERVISOR

Table 12 SPI_MCAL_SUPERVISOR

Name	SPI_MCAL_SUPERVISOR
Description	Supervisor Mode *Note: The macro is not user configurable.*
Verification method	The macro is generated always with value '0'.

Example(s)	**Action**	**Generated output**
	Generate 'Spi_Cfg.h'	`#define SPI_MCAL_SUPERVISOR (0)`

1.1.13 Macro: SPI_MCAL_USER1

Table 13 SPI_MCAL_USER1

Name	SPI_MCAL_USER1	
Description	User Mode *Note: The macro is not user configurable.*	
Verification method	The macro is generated always with value '1'.	
Example(s)	**Action**	**Generated output**
	Generate 'Spi_Cfg.h'	`#define SPI_MCAL_USER1 (1)`

1.1.14 Macro: SPI_INIT_CHECK_API

Table 14 SPI_INIT_CHECK_API

Name	SPI_INIT_CHECK_API	
Description	Enables/disables SpiInitCheckApi API	
Verification method	The macro is generated as STD_ON if SpiInitCheckApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	SpiInitCheckApi = True	`#define SPI_INIT_CHECK_API (STD_ON)`
	SpiInitCheckApi = False	`#define SPI_INIT_CHECK_API (STD_OFF)`

1.1.15 Macro: SPI_RUN_TIME_API_MODE

Table 15 SPI_RUN_TIME_API_MODE

Name	SPI_RUN_TIME_API_MODE	
Description	Decides the mode of execution of Run Time API's	
Verification method	The macro is generated as SPI_MCAL_USER1 if SpiRuntimeApiMode configuration parameter is set to 'SPI_MCAL_USER1' else the macro is generated as SPI_MCAL_SUPERVISOR.	
Example(s)	**Action**	**Generated output**
	SpiRuntimeApiMode = SPI_MCAL_USER1	`#define SPI_RUN_TIME_API_MODE (SPI_MCAL_USER1)`
	SpiRuntimeApiMode = SPI_MCAL_SUPERVISOR	`#define SPI_RUN_TIME_API_MODE (SPI_MCAL_SUPERVISOR)`

1.1.16 Macro: SPI_INIT_DEINIT_API_MODE

Table 16 SPI_INIT_DEINIT_API_MODE

Name	SPI_INIT_DEINIT_API_MODE	
Description	Decides the mode of execution of Init and DeInit API's.	
Verification method	The macro is generated as SPI_MCAL_USER1 if SpiInitDeInitApiMode configuration parameter is set to 'SPI_MCAL_USER1' else the macro is generated as SPI_MCAL_SUPERVISOR.	
Example(s)	**Action**	**Generated output**
	SpiInitDeInitApiMode = SPI_MCAL_USER1	`#define SPI_INIT_DEINIT_API_MODE (SPI_MCAL_USER1)`
	SpiInitDeInitApiMode = SPI_MCAL_SUPERVISOR	`#define SPI_INIT_DEINIT_API_MODE (SPI_MCAL_SUPERVISOR)`

1.1.17 Macro: SPI_DEV_ERROR_DETECT

Table 17 SPI_DEV_ERROR_DETECT

Name	SPI_DEV_ERROR_DETECT	
Description	Enables/disables the Development Error Detection.	
Verification method	The macro is generated as STD_ON if SpiDevErrorDetect configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	SpiDevErrorDetect= True	`#define SPI_DEV_ERROR_DETECT (STD_ON)`
	SpiDevErrorDetect= False	`#define SPI_DEV_ERROR_DETECT (STD_OFF)`

1.1.18 Macro: SPI_MULTICORE_ERROR_DETECT

Table 18 SPI_MULTICORE_ERROR_DETECT

Name	SPI_MULTICORE_ERROR_DETECT	
Description	Enables/disables MultiCore DET Check	
Verification method	The macro is generated as STD_ON if SpiMultiCoreErrorDetect configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	SpiMultiCoreErrorDetect = True	`#define SPI_MULTICORE_ERROR_DETECT (STD_ON)`
	SpiMultiCoreErrorDetect = False	`#define SPI_MULTICORE_ERROR_DETECT (STD_OFF)`

1.1.19 Macro: SPI_RUNTIME_ERROR_DETECT

Table 19 SPI_RUNTIME_ERROR_DETECT

Name	SPI_RUNTIME_ERROR_DETECT	
Description	Enables/disables runtime DET Check	
Verification method	The macro is generated as STD_ON if SpiRunTimeErrorDetect configuration parameter is set to 'True' else the macro is generated as STD_OFF for AUTOSAR version 4.4.0.	
Example(s)	**Action**	**Generated output**
	SpiRunTimeErrorDetect = True	`#define SPI_RUNTIME_ERROR_DETECT (STD_ON)`
	SpiRunTimeErrorDetect = False	`#define SPI_RUNTIME_ERROR_DETECT (STD_OFF)`

1.1.20 Macro: SPI_LEVEL_DELIVERED

Table 20 SPI_LEVEL_DELIVERED

Name	SPI_LEVEL_DELIVERED	
Description	Represents the LEVEL in which SPI operates.	
Verification method	macro is generated as • '0' if SpiLevelDelivered is set '0' • '1' if SpiLevelDelivered is set '1' • '2' if SpiLevelDelivered is set '2'	
Example(s)	**Action**	**Generated output**
	SpiLevelDelivered = 0	`#define SPI_LEVEL_DELIVERED (0)`
	SpiLevelDelivered = 1	`#define SPI_LEVEL_DELIVERED (1)`
	SpiLevelDelivered = 2	`#define SPI_LEVEL_DELIVERED (2)`

1.1.21 Macro: SPI_MAIN_FUNCTION_PERIOD

Table 21 SPI_MAIN_FUNCTION_PERIOD

Name	SPI_MAIN_FUNCTION_PERIOD	
Description	Specifies the Main Function handling period.	
Verification method	The macro is generated based on the value assigned to SpiMainFunctionPeriod configuration parameter.	
Example(s)	**Action**	**Generated output**
	SpiMainFunctionPeriod = 1.0E-4	`#define SPI_MAIN_FUNCTION_PERIOD (1.0E-4)`
	SpiMainFunctionPeriod = 1.0E-2	`#define SPI_MAIN_FUNCTION_PERIOD (1.0E-2)`

1.1.22 Macro: SPI_CHANNEL_BUFFERS_ALLOWED

Table 22 SPI_CHANNEL_BUFFERS_ALLOWED

Name	SPI_CHANNEL_BUFFERS_ALLOWED	
Description	Represents the allowed channel buffer type like Internal buffer (IB) or External buffer (EB) or Both.	
Verification method	The macro is generated based on the value assigned to SpiChannelBuffersAllowed configuration parameter.	
Example(s)	**Action**	**Generated output**
	SpiChannelBuffersAllowed= 0	`#define SPI_CHANNEL_BUFFERS_ALLOWED (0)`
	SpiChannelBuffersAllowed= 1	`#define SPI_CHANNEL_BUFFERS_ALLOWED (1)`
	SpiChannelBuffersAllowed= 2	`#define SPI_CHANNEL_BUFFERS_ALLOWED (2)`

1.1.23 Macro: SPI_CANCEL_API

Table 23 SPI_CANCEL_API

Name	SPI_CANCEL_API	
Description	Enables/disables Spi_Cancel API	
Verification method	The macro is generated as STD_ON if SpiCancelApi configuration parameter is set to ‘True’ else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	SpiCancelApi = True	`#define SPI_CANCEL_API (STD_ON)`
	SpiCancelApi = False	`#define SPI_CANCEL_API (STD_OFF)`

1.1.24 Macro: SPI_HW_STATUS_API

Table 24 SPI_HW_STATUS_API

Name	SPI_HW_STATUS_API	
Description	Enables/disables Spi_GetHWUnitStatus API	
Verification method	The macro is generated as STD_ON if SpiHwStatusApi configuration parameter is set to ‘True’ else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	SpiHwStatusApi = True	`#define SPI_HW_STATUS_API (STD_ON)`
	SpiHwStatusApi = False	`#define SPI_HW_STATUS_API (STD_OFF)`

1.1.25 Macro: SPI_CONTROL_LOOPBACK_API

Table 25 SPI_CONTROL_LOOPBACK_API

Name	SPI_CONTROL_LOOPBACK_API	
Description	Enables/disables Spi_ControlLoopBack API	
Verification method	The macro is generated as STD_ON if SpiEnableLoopBackApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	SpiEnableLoopBackApi = True	`#define SPI_CONTROL_LOOPBACK_API (STD_ON)`
	SpiEnableLoopBackApi = False	`#define SPI_CONTROL_LOOPBACK_API (STD_OFF)`

1.1.26 Macro: SPI_VERSION_INFO_API

Table 26 SPI_VERSION_INFO_API

Name	SPI_VERSION_INFO_API	
Description	Enables/disables Spi_GetVersionInfo API	
Verification method	The macro is generated as STD_ON if SpiVersionInfoApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	SpiVersionInfoApi = True	`#define SPI_VERSION_INFO_API (STD_ON)`
	SpiVersionInfoApi = False	`#define SPI_VERSION_INFO_API (STD_OFF)`

1.1.27 Macro: SPI_INTERRUPTIBLE_SEQ_ALLOWED

Table 27 SPI_INTERRUPTIBLE_SEQ_ALLOWED

Name	SPI_INTERRUPTIBLE_SEQ_ALLOWED	
Description	Enables/disables the interruptible feature.	
Verification method	The macro is generated as STD_ON if configuration parameter SpiLevelDelivered is not set to '0' and SpiInterruptibleSeqAllowed is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	SpiLevelDelivered !=0 and SpiInterruptibleSeqAllowed = True	`#define SPI_INTERRUPTIBLE_SEQ_ALLOWED (STD_ON)`
	SpiLevelDelivered !=0 and SpiInterruptibleSeqAllowed = False	`#define SPI_INTERRUPTIBLE_SEQ_ALLOWED`

		(STD_OFF)
	SpiLevelDelivered = 0	`#define SPI_INTERRUPTIBLE_SEQ_ALLOWED (STD_OFF)`

1.1.28 Macro: SPI_SEQ_INT_FALSE

Table 28 SPI_SEQ_INT_FALSE

Name	SPI_SEQ_INT_FALSE	
Description	Sequence is not interruptible. *Note: The macro is not user configurable.*	
Verification method	The macro is generated always with value '0'.	
Example(s)	**Action**	**Generated output**
	Generate 'Spi_Cfg.h'	`#define SPI_SEQ_INT_FALSE (0)`

1.1.29 Macro: SPI_SEQ_INT_TRUE

Table 29 SPI_SEQ_INT_TRUE

Name	SPI_SEQ_INT_TRUE	
Description	Sequence is interruptiable. *Note: The macro is not user configurable.*	
Verification method	The macro is generated always with value '1'.	
Example(s)	**Action**	**Generated output**
	Generate 'Spi_Cfg.h'	`#define SPI_SEQ_INT_TRUE (1)`

1.1.30 Macro: SPI_SUPPORT_CONCURRENT_SYNC_TRANSMIT

Table 30 SPI_SUPPORT_CONCURRENT_SYNC_TRANSMIT

Name	SPI_SUPPORT_CONCURRENT_SYNC_TRANSMIT	
Description	Enables/disables the concurrent transmission	
Verification method	The macro is generated as STD_ON if configuration parameter SpiLevelDelivered is not set to '1' and SpiSupportConcurrentSyncTransmit is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	SpiLevelDelivered !=1 and SpiSupportConcurrentSyncTransmit = True	`#define SPI_SUPPORT_CONCURRENT_SYNC_TRANSMIT (STD_ON)`
	SpiLevelDelivered !=1 and SpiSupportConcurrentSyncTransmit	`#define SPI_SUPPORT_CONCURRENT_SYNC_TRANSMIT`

	= False	`(STD_OFF)`
	SpiLevelDelivered =1	`#define SPI_SUPPORT_CONCURRENT_SYNC_TRANSMIT (STD_OFF)`

1.1.31 Macro: SPI_MAX_HW_UNIT

Table 31 SPI_MAX_HW_UNIT

Name	SPI_MAX_HW_UNIT	
Description	Maximum QSPI HW unit ID that is available for the specific device.	
Verification method	The macro is generated based on the maximum QSPI kernel Id that is available.The macro value is generated after incrementing the maximum kernel id value by 1.	
Example(s)	**Action**	**Generated output**
	If the kernels available for the device are QSPI0, QSPI1, QSPI2	`#define SPI_MAX_HW_UNIT (3)`
	If the kernels available for the device are QSPI0, QSPI1, QSPI3	`#define SPI_MAX_HW_UNIT (4)`

1.1.32 Macro: SPI_SYNC_BUS

Table 32 SPI_SYNC_BUS

Name	SPI_SYNC_BUS	
Description	QSPI HW is configured for Synchronous communication. *Note: The macro is not user configurable.*	
Verification method	The macro is generated always with value ‘0’.	
Example(s)	**Action**	**Generated output**
	Generate ‘Spi_Cfg.h’	`#define SPI_SYNC_BUS (0)`

1.1.33 Macro: SPI_ASYNC_BUS

Table 33 SPI_ASYNC_BUS

Name	SPI_ASYNC_BUS	
Description	QSPI HW is configured for Asynchronous communication. *Note: The macro is not user configurable.*	
Verification method	The macro is generated always with value ‘1’.	
Example(s)	**Action**	**Generated output**

	Generate ‘Spi_Cfg.h’	`#define SPI_ASYNC_BUS (1)`

1.1.34 Macro: SPI_HW_QSPIx_USED

Table 34 **SPI_HW_QSPIx_USED**

Name	SPI_HW_QSPIx_USED	
Description	Specifies if a particular QSPI is configured for communication.	
Verification method	The macro is generated as STD_ON based on the SpiHwUnit configured. If a QSPI is not configured, macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**

	SpiExternalDevice0/SpiHwUnit = QSPI0	`#define SPI_HW_QSPI0_USED (STD_ON)`
		`#define SPI_HW_QSPI1_USED (STD_OFF)`
		`#define SPI_HW_QSPI2_USED (STD_OFF)`
		`#define SPI_HW_QSPI3_USED (STD_OFF)`
		`#define SPI_HW_QSPI4_USED (STD_OFF)`
		`#define SPI_HW_QSPI5_USED (STD_OFF)`

	SpiExternalDevice0/SpiHwUnit = QSPI1	`#define SPI_HW_QSPI0_USED (STD_OFF)`
		`#define SPI_HW_QSPI1_USED (STD_ON)`
		`#define SPI_HW_QSPI2_USED (STD_OFF)`
		`#define SPI_HW_QSPI3_USED (STD_OFF)`
		`#define SPI_HW_QSPI4_USED (STD_OFF)`
		`#define SPI_HW_QSPI5_USED (STD_OFF)`

	SpiExternalDevice0/SpiHwUnit = QSPI0
	SpiExternalDevice1/SpiHwUnit = QSPI1
		`#define SPI_HW_QSPI0_USED (STD_ON)`
		`#define SPI_HW_QSPI1_USED (STD_ON)`
		`#define SPI_HW_QSPI2_USED (STD_OFF)`
		`#define SPI_HW_QSPI3_USED (STD_OFF)`
		`#define SPI_HW_QSPI4_USED (STD_OFF)`
		`#define SPI_HW_QSPI5_USED (STD_OFF)`

1.1.35 Macro: SPI_QSPIx_INDEX

Table 35 **SPI_QSPIx_INDEX**

Name	SPI_QSPIx_INDEX	
Description	Specifies the index for a QSPI HW *Note: The macro is not user configurable.*	
Verification method	The macro is generated if a QSPI is configured; else the macro is not generated.	
Example(s)	**Action**	**Generated output**

	QSPI0 is configured
	QSPI1 is configured
	QSPI2 is configured
	QSPI3 is configured
	QSPI4 is configured
	QSPI5 is configured
		`#define SPI_QSPI0_INDEX (0)`
		`#define SPI_QSPI1_INDEX (1)`
		`#define SPI_QSPI2_INDEX (2)`
		`#define SPI_QSPI3_INDEX (3)`
		`#define SPI_QSPI4_INDEX (4)`
		`#define SPI_QSPI5_INDEX (5)`

	QSPI0 is not configured QSPI1 is not configured QSPI2 is not configured QSPI3 is not configured QSPI4 is not configured QSPI5 is not configured	`The macro is not generated.`

1.1.36 Macro: SPI_QSPIx_HWTYPE

Table 36 SPI_QSPIx_HWTYPE

<table>
<tr><td><b>Name</b></td><td colspan="2">SPI_QSPI0_HWTYPE</td></tr>
<tr><td><b>Description</b></td><td colspan="2">Specifies if a particular QSPI is configured for Synchronous communication or Asynchronous communication</td></tr>
<tr><td><b>Verification method</b></td><td colspan="2">If a QSPI is configured
• If SpiLevelDelivered is set ‘2’, the macro is generated as SPI_ASYNC_BUS or SPI_SYNC_BUS based on the ‘SpiHwUnitSynchronous’ configuration parameter.
• If SpiLevelDelivered is set ‘1’, the macro is generated as SPI_ASYNC_BUS.
• If SpiLevelDelivered is set ‘0’, the macro is generated as SPI_SYNC_BUS.
If a QSPI is not configured, macro is not generated.</td></tr>
<tr><td><b>Example(s)</b></td><td><b>Action</b></td><td><b>Generated output</b></td></tr>
<tr><td></td><td>SpiLevelDelivered = 2
SpiExternalDevice0/SpiHwUnit = QSPI0
SpiJob0/ SpiHwUnitSynchronous = ASYNCHRONOUS</td><td><code>#define SPI_QSPI0_HWTYPE (SPI_ASYNC_BUS)</code></td></tr>
<tr><td></td><td>SpiLevelDelivered = 2
SpiExternalDevice0/SpiHwUnit = QSPI1
SpiJob0/ SpiHwUnitSynchronous = ASYNCHRONOUS</td><td><code>#define SPI_QSPI1_HWTYPE (SPI_ASYNC_BUS)</code></td></tr>
<tr><td></td><td>SpiLevelDelivered = 2
SpiExternalDevice0/SpiHwUnit = QSPI0
SpiJob0/ SpiHwUnitSynchronous = ASYNCHRONOUS
SpiExternalDevice1/SpiHwUnit =</td><td><code>#define SPI_QSPI0_HWTYPE (SPI_ASYNC_BUS)</code>
<code>#define SPI_QSPI1_HWTYPE (SPI_SYNC_BUS)</code></td></tr>
</table>

	QSPI1 SpiJob1/ SpiHwUnitSynchronous = SYNCHRONOUS	

	SpiLevelDelivered = 0
	SpiExternalDevice0/SpiHwUnit = QSPI0
	SpiExternalDevice1/SpiHwUnit = QSPI1
		`#define SPI_QSPI0_HWTYPE (SPI_SYNC_BUS)`
		`#define SPI_QSPI1_HWTYPE (SPI_SYNC_BUS)`

	SpiLevelDelivered = 1
	SpiExternalDevice0/SpiHwUnit = QSPI0
	SpiExternalDevice1/SpiHwUnit = QSPI1
		`#define SPI_QSPI0_HWTYPE (SPI_ASYNC_BUS)`
		`#define SPI_QSPI1_HWTYPE (SPI_ASYNC_BUS)`

1.1.37 Macro: SPI_DELAY_TIMEOUT

Table 37 SPI_DELAY_TIMEOUT

Name	SPI_DELAY_TIMEOUT	
Description	QSPI delay timeout value *Note: The macro is not user configurable.*	
Verification method	The macro is generated always with value ‘0xFFFFFFFE’.	
Example(s)	**Action**	**Generated output**
	Generate ‘Spi_Cfg.h’	`#define SPI_DELAY_TIMEOUT (0xFFFFFFFE)`

1.1.38 Macro: SPI_CLK_SLEEP_DISABLE

Table 38 SPI_CLK_SLEEP_DISABLE

Name	SPI_CLK_SLEEP_DISABLE	
Description	QSPI Sleep Disable. *Note: The macro is not user configurable.*	
Verification method	The macro is generated always with value ‘0x00000008’.	
Example(s)	**Action**	**Generated output**
	Generate ‘Spi_Cfg.h’	`#define SPI_CLK_SLEEP_DISABLE (0x00000008)`

1.1.39 Macro: SPI_CLK_SLEEP_ENABLE

Table 39 SPI_CLK_SLEEP_ENABLE

Name	SPI_CLK_SLEEP_ENABLE	
Description	QSPI Sleep Enable. *Note: The macro is not user configurable.*	
Verification method	The macro is generated always with value ‘0’.	
Example(s)	**Action**	**Generated output**
	Generate ‘Spi_Cfg.h’	`#define SPI_CLK_SLEEP_ENABLE (0x00000000)`

1.1.40 Macro: SPI_JOB_DELIMITER

Table 40 SPI_JOB_DELIMITER

Name	SPI_JOB_DELIMITER	
Description	QSPI Job Delimiter used to specify the end of job processing. *Note: The macro is not user configurable.*	
Verification method	The macro is generated always with value ‘0xFFFF’.	
Example(s)	**Action**	**Generated output**
	Generate ‘Spi_Cfg.h’	`#define SPI_JOB_DELIMITER (0xFFFF)`

1.1.41 Macro: SPI_SEQUENCE_DELIMITER

Table 41 SPI_SEQUENCE_DELIMITER

Name	SPI_SEQUENCE_DELIMITER	
Description	QSPI Sequence Delimiter used to specify the end of sequence processing. *Note: The macro is not user configurable.*	
Verification method	The macro is generated always with value ‘0xFF’.	
Example(s)	**Action**	**Generated output**
	Generate ‘Spi_Cfg.h’	`#define SPI_SEQUENCE_DELIMITER (0xFF)`

1.1.42 Macro: SPI_CHANNEL_DELIMITER

Table 42 SPI_CHANNEL_DELIMITER

Name	SPI_CHANNEL_DELIMITER
Description	QSPI Channel Delimiter used to specify the end of Channel processing.

	Note: The macro is not user configurable.	
Verification method	The macro is generated always with value '0xFF'.	
Example(s)	**Action**	**Generated output**
	Generate 'Spi_Cfg.h'	`#define SPI_CHANNEL_DELIMITER (0xFF)`

1.1.43 Macro: SPI_QSPI_HW_DELIMITER

Table 43 SPI_QSPI_HW_DELIMITER

Name	SPI_QSPI_HW_DELIMITER	
Description	QSPI HW Delimiter used to specify the end of QSPI HW. *Note: The macro is not user configurable.*	
Verification method	The macro is generated always with value '0xFF'.	
Example(s)	**Action**	**Generated output**
	Generate 'Spi_Cfg.h'	`#define SPI_QSPI_HW_DELIMITER (0xFF)`

1.1.44 Macro: SPI_IB_BUFFER_SIZE_COREx

Table 44 SPI_IB_BUFFER_SIZE_COREx

Name	SPI_IB_BUFFER_SIZE_COREx	
Description	Specifies the total IB buffer size required for a core. Generation of macro always ensures that the size is word aligned.	
Verification method	If a QSPI HW is assinged to a core (Applicable in AUTOSAR 4.2.2, when QSPI HW is asynchronous): - If 'SpiChannelType= IB', the macro is genrated as sum of all the IB buffer size calculated from all the IB channels allocated to the core. - If 'SpiChannelType= EB', the macro is genrated as '0'. If a Core is not assigned with QSPI , the macro is not generated. If a QSPI HW is synchronous in case of AUTOSAR 4.2.2, then th macro is genrated as '0'. If the total buffer size calculated is more than 65535 then an error is reported.	
Example(s)	**Action**	**Generated output**
	SpiLevelDelivered = 1 QSPI0 – Resource allocated to Core0 SpiExternalDevice0/SpiHwUnit = QSPI0 (Applicable for AUTOSAR 4.2.2 only) SpiJob0/	`#define SPI_IB_BUFFER_SIZE_CORE0 (36)`

SpiHwUnitSynchronous = ASYNCHRONOUS All channels SpiChannelType = IB SpiChannel_0/SpiIbNBuffers =10 SpiChannel_0/SpiDataWidth =8 SpiChannel_1/SpiIbNBuffers =10 SpiChannel_1/SpiDataWidth =8 SpiChannel_2/SpiIbNBuffers =10 SpiChannel_2/SpiDataWidth =8	
SpiLevelDelivered = 1 SpiExternalDevice0/SpiHwUnit = QSPI1 QSPI1 – Resource allocated to Core1 SpiJob0/ SpiHwUnitSynchronous(Applicable for AUTOSAR 4.2.2 only) = ASYNCHRONOUS All channels SpiChannelType = IB SpiChannel_0/SpiIbNBuffers =10 SpiChannel_0/SpiDataWidth =8 SpiChannel_1/SpiIbNBuffers =10 SpiChannel_1/SpiDataWidth =8 SpiChannel_2/SpiIbNBuffers =10 SpiChannel_2/SpiDataWidth =8	#define SPI_IB_BUFFER_SIZE_CORE1 (36)
SpiLevelDelivered = 1 SpiExternalDevice0/SpiHwUnit = QSPI0 QSPI0- Resource allocated to Core0 SpiJob0/ SpiHwUnitSynchronous(Applicable for AUTOSAR 4.2.2 only) = ASYNCHRONOUS All channels SpiChannelType = IB SpiChannel_0/SpiIbNBuffers =10 SpiChannel_0/SpiDataWidth =8 SpiChannel_1/SpiIbNBuffers =10 SpiChannel_1/SpiDataWidth =16 SpiChannel_2/SpiIbNBuffers =10 SpiChannel_2/SpiDataWidth =32	#define SPI_IB_BUFFER_SIZE_CORE1 (72)
SpiLevelDelivered = 2 SpiExternalDevice1/SpiHwUnit = QSPI0	#define SPI_IB_BUFFER_SIZE_CORE0 (40) #define SPI_IB_BUFFER_SIZE_CORE1

	QSPI0- Resource allocated to Core0 SpiJob1/ SpiHwUnitSynchronous = ASYNCHRONOUS SpiChannel_0/SpiIbNBuffers =10 SpiChannel_0/SpiDataWidth =32 SpiExternalDevice0/SpiHwUnit = QSPI1 QSPI1- Resource Allocated to Core1 SpiJob0/ SpiHwUnitSynchronous(Applicable for AUTOSAR 4.2.2 only) = ASYNCHRONOUS SpiChannel_0/SpiIbNBuffers =10 SpiChannel_0/SpiDataWidth =8 SpiChannel_1/SpiIbNBuffers =10 SpiChannel_1/SpiDataWidth =16	`(32)`
	SpiLevelDelivered = 1 SpiExternalDevice0/SpiHwUnit = QSPI0 QSPI0- Resource allocated to Core0 SpiJob0/ SpiHwUnitSynchronous(Applicable for AUTOSAR 4.2.2 only) = ASYNCHRONOUS All channels SpiChannelType = EB SpiChannel_0/SpiDataWidth =8 SpiChannel_1/SpiDataWidth =8 SpiChannel_2/SpiDataWidth =8	`#define SPI_IB_BUFFER_SIZE_CORE0 (0)`
	SpiLevelDelivered = 0 SpiExternalDevice0/SpiHwUnit = QSPI0 QSPI0- Resource allocated to Core0 All channels SpiChannelType = IB SpiChannel_0/SpiDataWidth =8 SpiChannel_1/SpiDataWidth =8 SpiChannel_2/SpiDataWidth =8	`#define SPI_IB_BUFFER_SIZE_CORE0 (0)`

1.1.45 Macro: SPI_JOB_QUEUE_LENGTH_QSPIx

Table 45 SPI_JOB_QUEUE_LENGTH_QSPIx

Name	SPI_JOB_QUEUE_LENGTH_QSPIx

Description	Specifies the Job queue length for a QSPI	
Verification method	If a QSPI is configured, the macro is generated based on the value of configuration parameter 'SpiJobQueueLengthQspix+1' else the macro is set '0'	
Example(s)	**Action**	**Generated output**

	SpiHwConfigurationQspi_0/S SpiHwConfigKernel = QSPI0
	SpiJobQueueLengthQspix = 100
	No other QSPI HW is Configured.
		`#define SPI_JOB_QUEUE_LENGTH_QSPI0 (101)`
		`#define SPI_JOB_QUEUE_LENGTH_QSPI1 (0)`
		`#define SPI_JOB_QUEUE_LENGTH_QSPI2 (0)`
		`#define SPI_JOB_QUEUE_LENGTH_QSPI3 (0)`
		`#define SPI_JOB_QUEUE_LENGTH_QSPI4 (0)`
		`#define SPI_JOB_QUEUE_LENGTH_QSPI5 (0)`

	SpiHwConfigurationQspi_0/S SpiHwConfigKernel = QSPI1
	SpiJobQueueLengthQspix = 10
	No other QSPI HW is Configured.
		`#define SPI_JOB_QUEUE_LENGTH_QSPI0 (0)`
		`#define SPI_JOB_QUEUE_LENGTH_QSPI1 (11)`
		`#define SPI_JOB_QUEUE_LENGTH_QSPI2 (0)`
		`#define SPI_JOB_QUEUE_LENGTH_QSPI3 (0)`
		`#define SPI_JOB_QUEUE_LENGTH_QSPI4 (0)`
		`#define SPI_JOB_QUEUE_LENGTH_QSPI5 (0)`

	SpiHwConfigurationQspi_0/S SpiHwConfigKernel = QSPI0
	SpiJobQueueLengthQspix = 10
	SpiHwConfigurationQspi_1/S SpiHwConfigKernel = QSPI1
	SpiJobQueueLengthQspix = 20
	No other QSPI HW is Configured.
		`#define SPI_JOB_QUEUE_LENGTH_QSPI0 (11)`
		`#define SPI_JOB_QUEUE_LENGTH_QSPI1 (21)`
		`#define SPI_JOB_QUEUE_LENGTH_QSPI2 (0)`
		`#define SPI_JOB_QUEUE_LENGTH_QSPI3 (0)`
		`#define SPI_JOB_QUEUE_LENGTH_QSPI4 (0)`
		`#define SPI_JOB_QUEUE_LENGTH_QSPI5 (0)`

1.1.46 Macro: SPI_QSPI_CHANNELx

Table 46 SPI_QSPI_CHANNELx

Name	SPI_QSPI_CHANNELx	
Description	Specifies the QSPI HW channel number. *Note: The macro is not user configurable and 'x' varies from 0 to 15.*	
Verification method	The macro is generated with values from 0 to 15.	
Example(s)	**Action**	**Generated output**

	Generate 'Spi_Cfg.h'	`#define SPI_QSPI_CHANNEL0 (0)`
		`#define SPI_QSPI_CHANNEL1 (1)`
		`#define SPI_QSPI_CHANNEL2 (2)`
		`#define SPI_QSPI_CHANNEL3 (3)`
		`#define SPI_QSPI_CHANNEL4 (4)`
		`#define SPI_QSPI_CHANNEL5 (5)`
		`#define SPI_QSPI_CHANNEL6 (6)`
		`#define SPI_QSPI_CHANNEL7 (7)`
		`#define SPI_QSPI_CHANNEL8 (8)`
		`#define SPI_QSPI_CHANNEL9 (9)`
		`#define SPI_QSPI_CHANNEL10 (10)`
		`#define SPI_QSPI_CHANNEL11(11)`
		`#define SPI_QSPI_CHANNEL12 (12)`
		`#define SPI_QSPI_CHANNEL13 (13)`
		`#define SPI_QSPI_CHANNEL14 (14)`
		`#define SPI_QSPI_CHANNEL15 (15)`

1.1.47 Macro: SPI_NUM_IB_CHANNELS_COREx

Table 47 SPI_NUM_IB_CHANNELS_COREx

Name	SPI_NUM_IB_CHANNELS_COREx	
Description	Total number of IB channels per Core.	
Verification method	If a Core is enabled, the macro is generated based on the total IB channels 'SpiChannelType= IB' falls under the core. If a Core not assigned with QSPI, the macro is not generated. If a core is enabled and there are no IB channels under the core, the macro is set '0'	
Example(s)	**Action**	**Generated output**
	SpiLevelDelivered = 1 QSPI0 – Resource allocated to Core0 SpiExternalDevice0/SpiHwUnit = QSPI0 All channels SpiChannelType = IB SpiChannel_0/SpiIbNBuffers =10	`#define SPI_NUM_IB_CHANNELS_CORE0 (3)`

SpiChannel_0/SpiDataWidth =8 SpiChannel_1/SpiIbNBuffers =10 SpiChannel_1/SpiDataWidth =8 SpiChannel_2/SpiIbNBuffers =10 SpiChannel_2/SpiDataWidth =8	
SpiLevelDelivered = 1 SpiExternalDevice0/SpiHwUnit = QSPI1 QSPI1 – Resource allocated to Core1 All channels SpiChannelType = IB SpiChannel_0/SpiIbNBuffers =10 SpiChannel_0/SpiDataWidth =8 SpiChannel_1/SpiIbNBuffers =10 SpiChannel_1/SpiDataWidth =8 SpiChannel_2/SpiIbNBuffers =10 SpiChannel_2/SpiDataWidth =8	`#define SPI_NUM_IB_CHANNELS_CORE1 (3)`
SpiLevelDelivered = 1 SpiExternalDevice0/SpiHwUnit = QSPI0 QSPI0- Resource allocated to Core0 All channels SpiChannelType = IB SpiChannel_0/SpiIbNBuffers =10 SpiChannel_0/SpiDataWidth =8 SpiChannel_1/SpiIbNBuffers =10 SpiChannel_1/SpiDataWidth =16 SpiChannel_2/SpiIbNBuffers =10 SpiChannel_2/SpiDataWidth =32	`#define SPI_NUM_IB_CHANNELS_CORE0 (3)`

SpiLevelDelivered = 2
SpiExternalDevice1/SpiHwUnit = QSPI0
QSPI0- Resource allocated to Core0
SpiChannel_0/SpiIbNBuffers =10
SpiChannel_0/SpiDataWidth =32
SpiExternalDevice0/SpiHwUnit = QSPI1
QSPI1- Resource Allocated to Core1
SpiChannel_0/SpiIbNBuffers =10
SpiChannel_0/SpiDataWidth =8
SpiChannel_1/SpiIbNBuffers =10
SpiChannel_1/SpiDataWidth =16
	`#define SPI_NUM_IB_CHANNELS_CORE0 (1)`
	`#define SPI_NUM_IB_CHANNELS_CORE1 (2)`

	SpiLevelDelivered = 1 SpiExternalDevice0/SpiHwUnit = QSPI0 QSPI0- Resource allocated to Core0 All channels SpiChannelType = EB SpiChannel_0/SpiDataWidth =8 SpiChannel_1/SpiDataWidth =8 SpiChannel_2/SpiDataWidth =8	`#define SPI_NUM_IB_CHANNELS_CORE0 (0)`

1.1.48 Macro: SPI_CORE<x>_ENABLE

Table 48 SPI_CORE<x>_ENABLE

Name	SPI_CORE<x>_ENABLE	
Description	Represents the Cores which are allocated with QSPI resources. *Note: ‘x’ varies from 0 to 5.*	
Verification method	The macro is generated as ‘STD_ON’ if a QSPI is assigned to a core else the macro is generated as ‘STD_OFF’	

Example(s)	QSPI0 – Assigned to Core0
	Other QSPI HW is not used.
		`#define SPI_CORE0_ENABLE (STD_ON)`
		`#define SPI_CORE1_ENABLE (STD_OFF)`
		`#define SPI_CORE2_ENABLE (STD_OFF)`
		`#define SPI_CORE3_ENABLE (STD_OFF)`
		`#define SPI_CORE4_ENABLE (STD_OFF)`
		`#define SPI_CORE5_ENABLE (STD_OFF)`

	QSPI0- Assigned to Core0
	QSPI1 – Assigned to Core0
	Other QSPI HW is not used.
		`#define SPI_CORE0_ENABLE (STD_ON)`
		`#define SPI_CORE1_ENABLE (STD_OFF)`
		`#define SPI_CORE2_ENABLE (STD_OFF)`
		`#define SPI_CORE3_ENABLE (STD_OFF)`
		`#define SPI_CORE4_ENABLE (STD_OFF)`
		`#define SPI_CORE5_ENABLE (STD_OFF)`

	QSPI0- Assigned to Core0
	QSPI1 – Assigned to Core1
	Other QSPI HW is not used.
		`#define SPI_CORE0_ENABLE (STD_ON)`
		`#define SPI_CORE1_ENABLE (STD_ON)`
		`#define SPI_CORE2_ENABLE (STD_OFF)`
		`#define SPI_CORE3_ENABLE (STD_OFF)`
		`#define SPI_CORE4_ENABLE (STD_OFF)`
		`#define SPI_CORE5_ENABLE (STD_OFF)`

1.1.49 Macro: SPI_NUM_EB_CHANNELS_COREx

Table 49 SPI_NUM_EB_CHANNELS_COREx

Name	SPI_NUM_EB_CHANNELS_COREx

Description	Total number of EB channels per Core.	
Verification method	If a Core is assigned with QSPI resources, the macro is generated based on the total EB channels 'SpiChannelType= EB' falls under the core. If a Core not assigned with QSPI, the macro is not generated. If a core is assigned with QSPI resources and there are no EB channels under the core, the macro is set '0'	
Example(s)	**Action**	**Generated output**
	SpiLevelDelivered = 1 QSPI0 – Resource allocated to Core0 SpiExternalDevice0/SpiHwUnit = QSPI0 All channels SpiChannelType = EB SpiChannel_0/SpiDataWidth =8 SpiChannel_1/SpiDataWidth =8 SpiChannel_2/SpiDataWidth =8	`#define SPI_NUM_EB_CHANNELS_CORE0 (3)`
	SpiLevelDelivered = 1 SpiExternalDevice0/SpiHwUnit = QSPI1 QSPI1 – Resource allocated to Core1 All channels SpiChannelType = EB SpiChannel_0/SpiDataWidth =8 SpiChannel_1/SpiDataWidth =8 SpiChannel_2/SpiDataWidth =8	`#define SPI_NUM_EB_CHANNELS_CORE1 (3)`
	SpiLevelDelivered = 1 SpiExternalDevice0/SpiHwUnit = QSPI0 QSPI0- Resource allocated to Core0 All channels SpiChannelType = EB SpiChannel_0/SpiDataWidth =8 SpiChannel_1/SpiDataWidth =16 SpiChannel_2/SpiDataWidth =32	`#define SPI_NUM_EB_CHANNELS_CORE0 (3)`

	SpiLevelDelivered = 2
	SpiExternalDevice1/SpiHwUnit = QSPI0
	QSPI0- Resource allocated to Core0
	SpiChannel_0/SpiDataWidth =32
	SpiExternalDevice0/SpiHwUnit =
		`#define SPI_NUM_EB_CHANNELS_CORE0 (1)`
		`#define SPI_NUM_EB_CHANNELS_CORE1 (2)`

	QSPI1 QSPI1- Resource Allocated to Core1 SpiChannel_0/SpiDataWidth =8 SpiChannel_1/SpiDataWidth =16	
	SpiLevelDelivered = 1 SpiExternalDevice0/SpiHwUnit = QSPI0 QSPI0- Resource allocated to Core0 All channels SpiChannelType = IB SpiChannel_0/SpiDataWidth =8 SpiChannel_1/SpiDataWidth =8 SpiChannel_2/SpiDataWidth =8	`#define SPI_NUM_EB_CHANNELS_CORE0 (0)`

1.1.50 Macro: SPI_DMA_MAX_TCS_NUM_QSPI<x>

Table 50 SPI_DMA_MAX_TCS_NUM_QSPI<x>

Name	SPI_DMA_MAX_TCS_NUM_QSPI<x>	
Description	DMA Transaction control set array size for a QSPI	
Verification method	If a QSPI is configured for Asynchronous communication, the macro is generated based on the Number of channels configured for a QSPI. If QSPI is not configured for Asynchronous communication, the macro is generated as '0'.	
Example(s)	SpiLevelDelivered = 1 SpiExternalDevice0/SpiHwUnit = QSPI0 SpiChannel_0 and SpiChannel_1 belongs to same job : SpiJob_0 SpiChannel_0/SpiDataWidth =8 SpiChannel_1/SpiDataWidth =8	`#define SPI_DMA_MAX_TCS_NUM_QSPI0 (2)`
	SpiLevelDelivered = 1 SpiExternalDevice0/SpiHwUnit = QSPI1 All channels belongs to same job: SpiJob_0 SpiChannel_0/SpiDataWidth =8 SpiChannel_1/SpiDataWidth =16 SpiChannel_2/SpiDataWidth =32	`#define SPI_DMA_MAX_TCS_NUM_QSPI1 (3)`

	SpiLevelDelivered = 1
	SpiExternalDevice0/SpiHwUnit = QSPI0
	SpiChannel_0 and SpiChannel_1 belongs to same job : SpiJob_0
	SpiChannel_0/SpiDataWidth =8
	SpiChannel_1/SpiDataWidth =8
	SpiLevelDelivered = 1
	SpiExternalDevice0/SpiHwUnit = QSPI1
	All channels belongs to same job: SpiJob_1
	SpiChannel_0/SpiDataWidth =8
	SpiChannel_1/SpiDataWidth =16
	SpiChannel_2/SpiDataWidth =32
		`#define SPI_DMA_MAX_TCS_NUM_QSPI0 (2)`
		`#define SPI_DMA_MAX_TCS_NUM_QSPI1 (3)`

1.1.51 Macro: SPI_CS_VIA_HW_OR_NONE

Table 51 SPI_CS_VIA_HW_OR_NONE

Name	SPI_CS_VIA_HW_OR_NONE	
Description	QSPI chip slect line is driven by Hardware. *Note: The macro is not user configurable.*	
Verification method	The macro is generated always with value ‘0xFFFF’.	
Example(s)	**Action**	**Generated output**
	Generate ‘Spi_Cfg.h’	`#define SPI_CS_VIA_HW_OR_NONE (0xFFFF)`

1.1.52 Macro: SPI_PARITY_EVEN

Table 52 SPI_PARITY_EVEN

Name	SPI_PARITY_EVEN
Description	Represents QSPI Even Parity *Note: The macro is not user configurable.*
Verification method	The macro is generated always with value ‘0x0’.

Example(s)	**Action**	**Generated output**
	Generate 'Spi_Cfg.h'	`#define SPI_PARITY_EVEN (0x0)`

1.1.53 Macro: SPI_PARITY_ODD

Table 53 SPI_PARITY_ODD

Name	SPI_PARITY_ODD	
Description	Represents QSPI Odd Parity *Note: The macro is not user configurable.*	
Verification method	The macro is generated always with value '0x1'.	
Example(s)	**Action**	**Generated output**
	Generate 'Spi_Cfg.h'	`#define SPI_PARITY_ODD (0x1)`

1.1.54 Macro: SPI_PARITY_UNUSED

Table 54 SPI_PARITY_UNUSED

Name	SPI_PARITY_UNUSED	
Description	Represents QSPI Unused Parity *Note: The macro is not user configurable.*	
Verification method	The macro is generated always with value '0x2'.	
Example(s)	**Action**	**Generated output**
	Generate 'Spi_Cfg.h'	`#define SPI_PARITY_UNUSED (0x2)`

1.1.55 Macro: SPI_EB_CHANNEL

Table 55 SPI_EB_CHANNEL

Name	SPI_EB_CHANNEL	
Description	Specifies EB(External buffer) channel *Note: The macro is not user configurable.*	
Verification method	The macro is generated always with value '0x0'.	
Example(s)	**Action**	**Generated output**
	Generate 'Spi_Cfg.h'	`#define SPI_EB_CHANNEL (0x0)`

1.1.56 Macro: SPI_IB_CHANNEL

Table 56 SPI_IB_CHANNEL

Name	SPI_IB_CHANNEL	
Description	Specifies IB(Internal buffer) channel *Note: The macro is not user configurable.*	
Verification method	The macro is generated always with value ‘0x1’.	
Example(s)	**Action**	**Generated output**
	Generate ‘Spi_Cfg.h’	`#define SPI_IB_CHANNEL (0x1)`

1.1.57 Macro: SPI_DMA_CHNL_INVALID

Table 57 SPI_DMA_CHNL_INVALID

Name	SPI_DMA_CHNL_INVALID	
Description	Specifies Invalid DMA channel. *Note: The macro is not user configurable.*	
Verification method	The macro is generated always with value ‘0xFF’.	
Example(s)	**Action**	**Generated output**
	Generate ‘Spi_Cfg.h’	`#define SPI_DMA_CHNL_INVALID (0xFF)`

1.1.58 Macro: SpiConf_SpiSequence_<Sequence_Name>

Table 58 SpiConf_SpiSequence_<Sequence_Name>

Name	SpiConf_SpiSequence_<Sequence_Name>	
Description	Symbolic name given for the sequence. *Note: <Sequence_Name> replaced by the ‘name given for sequence’ configured.*	
Verification method	The macro is generated with the value of Configuration parameter SpiSequenceId.	
Example(s)	**Action**	**Generated output**

	SpiSequenceId = 0
	SpiSequenceName = SpiSequence_0
		`#define SpiConf_SpiSequence_SpiSequence_0 (0)`

	SpiSequenceId = 1
	SpiSequenceName = SpiSequence_0
		`#define SpiConf_SpiSequence_SpiSequence_1 (0)`

	SpiSequenceId = 4
	SpiSequenceName = EEP_Test
		`#define SpiConf_SpiSequence_EEP_Test (4)`

1.1.59 Macro: SpiConf_SpiJob_<Job_Name>

Table 59 SpiConf_SpiJob_<Job_Name>

Name	SpiConf_SpiJob_<Job_Name>	
Description	Symbolic name given for the Job. *Note: <Job_Name> replaced by the 'name given for job' configured.*	
Verification method	The macro is generated with the value of Configuration parameter SpiJobId.	
Example(s)	**Action**	**Generated output**

	SpiJobId= 0
	SpiJobName = SpiJob_0
		`#define SpiConf_SpiJob_SpiJob_0 (0)`

	SpiJobId= 1
	SpiJobName = SpiJob_0
		`#define SpiConf_SpiJob_SpiJob_0 (1)`

	SpiJobId= 4
	SpiJobName = EEP_WRITE_TEST
		`#define SpiConf_SpiJob_EEP_WRITE_TEST (4)`

1.1.60 Macro: SpiConf_SpiChannel_<Channel_Name>

Table 60 SpiConf_SpiChannel_<Channel_Name>

Name	SpiConf_SpiChannel_<Channel_Name>	
Description	Symbolic name given for the Channel. *Note: <Channel_Name> replaced by the 'name given for channel' configured.*	
Verification method	The macro is generated with the value of Configuration parameter SpiChannelId.	
Example(s)	**Action**	**Generated output**

	SpiChannelId= 0
	SpiChannelName = SpiChannel_0
		`#define SpiConf_SpiChannel_SpiChannel_0 (0)`

	SpiChannelId= 1
	SpiChannelName = SpiChannel_0
		`#define SpiConf_SpiChannel_SpiChannel_0 (1)`

	SpiChannelId= 4
	SpiChannelName = EEP_ERASE_COMMAND
		`#define SpiConf_SpiChannel_EEP_ERASE_COMMAND (4)`

1.1.61 Macro: SPI_SEQUENCE_COUNT_CORE<x>

Table 61 SPI_SEQUENCE_COUNT_CORE<x>

Name	SPI_SEQUENCE_COUNT_CORE<x>
Description	Sequence count per Core.

Verification method	The macro is generated based on the number of sequences configured for a Core.	
Example(s)	**Action**	**Generated output**
	Configure SpiSequence_0 as follows. Sequence has one job SpiJob_0 SpiJob_0 drives the external device SpiExternalDevice_0 SpiExternalDevice_0/SpiHwUint = QSPI0 QSPI0 – allocated to Core0 No other kernels are configured.	#define SPI_SEQUENCE_COUNT_CORE0 (1) #define SPI_SEQUENCE_COUNT_CORE1 (0) #define SPI_SEQUENCE_COUNT_CORE2 (0) #define SPI_SEQUENCE_COUNT_CORE3 (0) #define SPI_SEQUENCE_COUNT_CORE4 (0) #define SPI_SEQUENCE_COUNT_CORE5 (0)
	Configure SpiSequence_0 as follows. Sequence has one job SpiJob_0 SpiJob_0 drives the external device SpiExternalDevice_0 SpiExternalDevice_0/SpiHwUint = QSPI0 QSPI0 – allocated to Core0 No other kernels are configured. Configure SpiSequence_1 as follows. Sequence has one job SpiJob_1 SpiJob_1 drives the external device SpiExternalDevice_1 SpiExternalDevice_1/SpiHwUint = QSPI1 QSPI1 – allocated to Core1	#define SPI_SEQUENCE_COUNT_CORE0 (1) #define SPI_SEQUENCE_COUNT_CORE1 (1) #define SPI_SEQUENCE_COUNT_CORE2 (0) #define SPI_SEQUENCE_COUNT_CORE3 (0) #define SPI_SEQUENCE_COUNT_CORE4 (0) #define SPI_SEQUENCE_COUNT_CORE5 (0)
	Configure SpiSequence_0 as follows. Sequence has one job SpiJob_0 SpiJob_0 drives the external device SpiExternalDevice_0 SpiExternalDevice_0/SpiHwUint = QSPI0 QSPI0 – allocated to Core0 No other kernels are configured. Configure SpiSequence_1 as follows. Sequence has one job SpiJob_1 SpiJob_1 drives the external	#define SPI_SEQUENCE_COUNT_CORE0 (2) #define SPI_SEQUENCE_COUNT_CORE1 (0) #define SPI_SEQUENCE_COUNT_CORE2 (0) #define SPI_SEQUENCE_COUNT_CORE3 (0) #define SPI_SEQUENCE_COUNT_CORE4 (0) #define SPI_SEQUENCE_COUNT_CORE5 (0)

	device SpiExternalDevice_1 SpiExternalDevice_1/SpiHwUint = QSPI1 QSPI1 – allocated to Core0	

	Configure SpiSequence_0 as follows.
	Sequence has one job SpiJob_0
	SpiJob_0 drives the external device SpiExternalDevice_0
	SpiExternalDevice_0/SpiHwUint = QSPI0
	QSPI0 – allocated to Core1
	No other kernels are configured.
	Configure SpiSequence_1 as follows.
	Sequence has one job SpiJob_1
	SpiJob_1 drives the external device SpiExternalDevice_1
	SpiExternalDevice_1/SpiHwUint = QSPI1
	QSPI1 – allocated to Core1
		`#define SPI_SEQUENCE_COUNT_CORE0 (0)`
		`#define SPI_SEQUENCE_COUNT_CORE1 (2)`
		`#define SPI_SEQUENCE_COUNT_CORE2 (0)`
		`#define SPI_SEQUENCE_COUNT_CORE3 (0)`
		`#define SPI_SEQUENCE_COUNT_CORE4 (0)`
		`#define SPI_SEQUENCE_COUNT_CORE5 (0)`

1.1.62 Macro: SPI_JOB_COUNT_CORE<x>

Table 62 SPI_JOB_COUNT_CORE<x>

Name	SPI_JOB_COUNT_CORE<x>	
Description	Job count per Core.	
Verification method	The macro is generated based on the number of Jobs configured for a Core.	
Example(s)	**Action**	**Generated output**

	Configure SpiSequence_0 as follows.
	Sequence has one job SpiJob_0
	SpiJob_0 drives the external device SpiExternalDevice_0
	SpiExternalDevice_0/SpiHwUint = QSPI0
	QSPI0 – allocated to Core0
	No other kernels are configured.
		`#define SPI_JOB_COUNT_CORE0 (1)`
		`#define SPI_JOB_COUNT_CORE1 (0)`
		`#define SPI_JOB_COUNT_CORE2 (0)`
		`#define SPI_JOB_COUNT_CORE3 (0)`
		`#define SPI_JOB_COUNT_CORE4 (0)`
		`#define SPI_JOB_COUNT_CORE5 (0)`

	Configure SpiSequence_0 as follows.
	Sequence has one job SpiJob_0
		`#define SPI_JOB_COUNT_CORE0 (1)`
		`#define SPI_JOB_COUNT_CORE1 (1)`
		`#define SPI_JOB_COUNT_CORE2 (0)`

SpiJob_0 drives the external device SpiExternalDevice_0 SpiExternalDevice_0/SpiHwUint = QSPI0 QSPI0 – allocated to Core0 No other kernels are configured. Configure SpiSequence_1 as follows. Sequence has one job SpiJob_1 SpiJob_1 drives the external device SpiExternalDevice_1 SpiExternalDevice_1/SpiHwUint = QSPI1 QSPI1 – allocated to Core1	#define SPI_JOB_COUNT_CORE3 (0) #define SPI_JOB_COUNT_CORE4 (0) #define SPI_JOB_COUNT_CORE5 (0)
Configure SpiSequence_0 as follows. Sequence has one job SpiJob_0 SpiJob_0 drives the external device SpiExternalDevice_0 SpiExternalDevice_0/SpiHwUint = QSPI0 QSPI0 – allocated to Core0 No other kernels are configured. Configure SpiSequence_1 as follows. Sequence has one job SpiJob_1 SpiJob_1 drives the external device SpiExternalDevice_1 SpiExternalDevice_1/SpiHwUint = QSPI1 QSPI1 – allocated to Core0	#define SPI_JOB_COUNT_CORE0 (2) #define SPI_JOB_COUNT_CORE1 (0) #define SPI_JOB_COUNT_CORE2 (0) #define SPI_JOB_COUNT_CORE3 (0) #define SPI_JOB_COUNT_CORE4 (0) #define SPI_JOB_COUNT_CORE5 (0)
Configure SpiSequence_0 as follows. Sequence has one job SpiJob_0 SpiJob_0 drives the external device SpiExternalDevice_0 SpiExternalDevice_0/SpiHwUint = QSPI0 QSPI0 – allocated to Core1 No other kernels are configured. Configure SpiSequence_1 as	#define SPI_JOB_COUNT_CORE0 (0) #define SPI_JOB_COUNT_CORE1 (2) #define SPI_JOB_COUNT_CORE2 (0) #define SPI_JOB_COUNT_CORE3 (0) #define SPI_JOB_COUNT_CORE4 (0) #define SPI_JOB_COUNT_CORE5 (0)

	follows. Sequence has one job SpiJob_1 SpiJob_1 drives the external device SpiExternalDevice_1 SpiExternalDevice_1/SpiHwUint = QSPI1 QSPI1 – allocated to Core1	

1.1.63 Macro: SPI_QSPI<x>_CORE

Table 63 SPI_QSPI<x>_CORE

Name	SPI_QSPI<x>_CORE	
Description	Core to which a QSPI resource is allocated.	
Verification method	The macro is generated based on the allocation of QSPI resource to a Core. The macro is not generated if a QSPI is not configured.	
Example(s)	**Action**	**Generated output**
	Allocate QSPI0 to Core0 No other QSPI HW is configured	`#define SPI_QSPI0_CORE (0)`
	Allocate QSPI0 to Core1 No other QSPI HW is configured	`#define SPI_QSPI0_CORE (1)`

	Allocate QSPI0 to Core0
	Allocate QSPI1 to Core0
	No other QSPI HW is configured
		`#define SPI_QSPI0_CORE (0)`
		`#define SPI_QSPI1_CORE (0)`

	Allocate QSPI0 to Core1
	Allocate QSPI1 to Core0
	No other QSPI HW is configured
		`#define SPI_QSPI0_CORE (1)`
		`#define SPI_QSPI1_CORE (0)`

1.2 File: Spi[_<variant>]_PBcfg.c

The generated source file contains all post-build configuration parameters. Post-build time configuration mechanism allows configurable functionality of SPI driver that is deployed as object code. The file is generated in 'src' folder.

1.2.1 Structure: Spi_Config[_<variant>]

Table 64 Spi_Config[_<variant>]

Name	Spi_Config[_<variant>]
Type	Spi_ConfigType
Description	Root configuration structure of SPI driver which will be used during initialization.

Verification method	The generated structure is present in Spi[_<variant>]_PBcfg.c file. The <variant> indicates the name of the post-build variant. For a variant-aware configuration the structure name is appended with the variant name. For variant-unaware configuration <variant> is ignored.	
Example(s)	**Action**	**Generated output**

	Configure 1 QSPI to Core0 (variant-unaware)	`const Spi_ConfigType Spi_Config =`
		`{`
		`  {`
		`    &Spi_Config_Core0,`
		`    NULL_PTR,`
		`    NULL_PTR,`
		`    NULL_PTR,`
		`    NULL_PTR,`
		`    NULL_PTR`
		`  },`
		`  SequenceLookupIndex,`
		`  JobLookupIndex,`
		`  ChannelLookupIndex,`
		`  /*Total number of Sequence*/`
		`  4U,`
		`  /*Total number of Jobs*/`
		`  5U,`
		`  /*Total number of Channels*/`
		`  15U,`
		`  /*Sync Delay*/`
		`  65535U`
		`};`

	Configure 1 QSPI to Core0 (variant-aware. Variant name is 'Petrol')	`const Spi_ConfigType`
		`Spi_Config_Petrol =`
		`{`
		`  {`
		`    &Spi_Config_Core0_Petrol,`
		`    NULL_PTR,`
		`    NULL_PTR,`
		`    NULL_PTR,`
		`    NULL_PTR,`
		`    NULL_PTR`
		`  },`
		`  SequenceLookupIndex_Petrol,`

		JobLookupIndex_Petrol, ChannelLookupIndex_Petrol, /*Total number of Sequence*/ 4U, /*Total number of Jobs*/ 5U, /*Total number of Channels*/ 15U, /*Sync Delay*/ 65535U };

1.2.1.1 Member: CoreConfigPtr[6]

Table 65 CoreConfigPtr[6]

Name	CoreConfigPtr[6]	
Type	Spi_CoreConfigType*	
Description	Array of core-specific configuration.	
Verification method	The generated structure member is present in the Spi_Config[_<variant>] structure. If a Core<x> is allocated at least one QSPI HW, then the element <x> is generated as '&Spi_Config_Core<x>[_<variant>]' else 'NULL_PTR' is generated. (x in range 0 to 5).	
Example(s)	**Action**	**Generated output**

	All the QSPI HW is allocated to Core 0 (variant-unaware)	{
		&Spi_Config_Core0,
		NULL_PTR,
		NULL_PTR,
		NULL_PTR,
		NULL_PTR,
		NULL_PTR
		}

	All the QSPI HW is allocated to Core 0 (variant-aware. Variant name is 'Petrol')	{ &Spi_Config_Core0_Petrol, NULL_PTR, NULL_PTR, NULL_PTR, NULL_PTR, NULL_PTR

		`}`

	All the QSPI HW is split between all cores except Core 0. (variant-unaware)	`{`
		`NULL_PTR,`
		`&Spi_Config_Core1,`
		`&Spi_Config_Core2,`
		`&Spi_Config_Core3,`
		`&Spi_Config_Core4,`
		`&Spi_Config_Core5`
		`}`

1.2.1.2 Member: SequenceLookup

Table 66 SequenceLookup

Name	SequenceLookup	
Type	uint8*	
Description	Reference for Sequence ID lookup table.	
Verification method	For a variant-aware configuration, Member name is appended with the variant name. For variant-unaware configuration <variant> is ignored.	
Example(s)	**Action**	**Generated output**
	variant-aware configuration(Variant name is ‘Petrol’)	`SequenceLookupIndex_Petrol`
	variant-unaware configuration	`SequenceLookupIndex`

1.2.1.3 Member: JobLookup

Table 67 JobLookup

Name	JobLookup	
Type	uint16*	
Description	Reference for Job ID lookup table.	
Verification method	For a variant-aware configuration, Member name is appended with the variant name. For variant-unaware configuration <variant> is ignored.	
Example(s)	**Action**	**Generated output**
	variant-aware configuration(Variant name is ‘Petrol’)	`JobLookupIndex_Petrol`
	variant-unaware configuration	`JobLookupIndex`

1.2.1.4 Member: ChannelLookup

Table 68 ChannelLookup

Name	ChannelLookup

Type	uint8*	
Description	Reference for Channel ID lookup table.	
Verification method	For a variant-aware configuration, Member name is appended with the variant name. For variant-unaware configuration <variant> is ignored.	
Example(s)	**Action**	**Generated output**
	variant-aware configuration(Variant name is ‘Petrol’)	`ChannelLookupIndex_Petrol`
	variant-unaware configuration	`ChannelLookupIndex`

1.2.1.5 Member: NoOfSequences

Table 69 NoOfSequences

Name	NoOfSequences	
Type	Spi_SequenceType	
Description	Total number of Sequences configured.	
Verification method	The value for the member is generated by counting all the configured sequences for container SpiDriver/SpiSequence.	
Example(s)	**Action**	**Generated output**
	Configure 10 sequences SpiDriver/SpiSequence/	`10`
	Configure 1 sequence SpiDriver/SpiSequence/	`1`

1.2.1.6 Member: NoOfJobs

Table 70 NoOfJobs

Name	NoOfJobs	
Type	Spi_JobType	
Description	Total number of Jobs configured.	
Verification method	The value for the member is generated by counting all the configured jobs for container SpiDriver/SpiJob.	
Example(s)	**Action**	**Generated output**
	Configure 10 jobs SpiDriver/SpiJob/	`10`
	Configure 1 job SpiDriver/SpiJob/	`1`

1.2.1.7 Member: NoOfChannels

Table 71 NoOfSequences

Name	NoOfChannels
Type	Spi_ChannelType

Description	Total number of Channels configured.	
Verification method	The value for the member is generated by counting all the configured channels for container SpiDriver/SpiChannel.	
Example(s)	**Action**	**Generated output**
	Configure 10 channels SpiDriver/SpiChannel/	`10`
	Configure 1 channel SpiDriver/SpiChannel/	`1`

1.2.1.8 Member: SyncTimeout

Table 72 SyncTimeout

Name	SyncTimeout	
Type	uint32	
Description	Timeout value used for Synchonrous communication.	
Verification method	The member is generated based on the value given for the configuration parameter SpiSyncTransmitTimeoutDuration.	
Example(s)	**Action**	**Generated output**
	SpiSyncTransmitTimeoutDuration = 0xFF	`255`
	SpiSyncTransmitTimeoutDuration = 0xFFFF	`65535`

1.2.2 Structure: Spi_Config_Core<x>[_<variant>]

Table 73 Spi_Config_Core<x>[_<variant>]

Name	Spi_Config_Core<x>[_<variant>]	
Type	Spi_CoreConfigType	
Description	Configuration structure of SPI driver for Core <x> which will be referenced in root configuration structure. (x ranges from 0 to 5)	
Verification method	The generated file has this structure if atleast one QSPI HW is assigned to Core <x>. <Variant> indicates the name of the post-build variant. For a variant aware configuration the structure name is appended with the variant name. For variant unaware configuration <variant> is ignored.	
Example(s)	**Action**	**Generated output**

	Configure 1 QSPI (QSPI0) and allocate to Core0 (variant-aware and variant name = Petrol)	`const Spi_CoreConfigType Spi_Config_Core0_Petrol =`
		`{`
		`  /* Sequence Configuration */`
		`  Spi_kSequenceConfig_Core0,`
		
		`  /* Job configuration */`

	Spi_kJobConfig_Core0,
	
	/* Channel Configuration */
	Spi_kChannelConfig_Core0,
	
	Spi_ChannelOffsets_Core0,
	/* QSPI Hw configuration */
	{
	&Spi_kQspiHwConfigQSPI0,
	NULL_PTR,
	NULL_PTR,
	NULL_PTR,
	NULL_PTR,
	NULL_PTR,
	},
	/* Hw Map Index */
	/*
	(000 QSPI not configured for core0)
	(001 QSPI configured as Sync for core0)
	(010 QSPI configured as Async for core0)
	QSPI5 - 0
	QSPI4 - 0
	QSPI3 - 0
	QSPI2 - 0
	QSPI1 - 1
	QSPI0 - 2
	*/
	0x0000aU,
	/* No. of Sequences configured */
	4U,
	/* No. of Jobs configured */
	5U,
	/* No. of Channels configured */
	15U
	};

Configure 1 QSPI(QSPI0) and allocate to Core0 (variant-unaware)	const Spi_CoreConfigType Spi_Config_Core0 =

```
{
  /* Sequence Configuration */
  Spi_kSequenceConfig_Core0,

  /* Job configuration */
  Spi_kJobConfig_Core0,

  /* Channel Configuration */
  Spi_kChannelConfig_Core0,

  Spi_ChannelOffsets_Core0,
  /* QSPI Hw configuration */
  {
    &Spi_kQspiHwConfigQSPI0,
    NULL_PTR,
    NULL_PTR,
    NULL_PTR,
    NULL_PTR,
    NULL_PTR,
  },
  /* Hw Map Index */
  /*
  (000 QSPI not configured for core0)
  (001 QSPI configured as Sync for
core0)
  (010 QSPI configured as Async for
core0)
  QSPI5 - 0
  QSPI4 - 0
  QSPI3 - 0
  QSPI2 - 0
  QSPI1 - 1
  QSPI0 - 2
  */
  0x0000aU,
  /* No. of Sequences configured */
  4U,
  /* No. of Jobs configured */
  5U,
```

		`  /* No. of Channels configured */`
		`  15U`
		`};`

1.2.2.1 Member: SequenceConfigPtr

Table 74 SequenceConfigPtr

Name	SequenceConfigPtr	
Type	Spi_SequenceConfigType*	
Description	Pointer to the base of array which stores the infomation of each Sequence configured to Core<x>.	
Verification method	The structure member is generated as Spi_kSequenceConfig_Core<x> (x ranges 0 to 5) reference to base address of array which stores the sequence information of Core <x>.	
Example(s)	**Action**	**Generated output**
	Configure atleast 1 QSPI HW to Core 0.	`Spi_kSequenceConfig_Core0`
	Configure atleast 1 QSPI HW to Core 1.	`Spi_kSequenceConfig_Core1`

1.2.2.2 Member: JobConfigPtr

Table 75 JobConfigPtr

Name	JobConfigPtr	
Type	Spi_JobConfigType*	
Description	Pointer to the base of array which stores the infomation of each Job configured to Core<x>.	
Verification method	The structure member is generated as Spi_kJobConfig_Core<x> (x ranges 0 to 5) reference base address of array which stores the job information of Core <x>.	
Example(s)	**Action**	**Generated output**
	Configure atleast 1 QSPI HW to Core0	`Spi_kJobConfig_Core0`
	Configure atleast 1 QSPI HW to Core1	`Spi_kJobConfig_Core1`

1.2.2.3 Member: ChannelConfigPtr

Table 76 ChannelConfigPtr

Name	ChannelConfigPtr
Type	Spi_ChannelConfigType*
Description	Pointer to the base of array which stores the infomation of each Channel configured to Core<x>.
Verification method	The structure member is generated as Spi_kChannelConfig_Core<x> (x ranges 0 to 5)

	reference base address of array which stores the channel information of Core <x>.	
Example(s)	**Action**	**Generated output**
	Configure atleast 1 QSPI HW to Core 0.	`Spi_kChannelConfig_Core0`
	Configure atleast 1 QSPI HW to Core 1.	`Spi_kChannelConfig_Core1`

1.2.2.4 Member: ChannelOffsetInfo

Table 77 ChannelOffsetInfo

Name	ChannelOffsetInfo	
Type	Spi_CoreChannelOffsetType*	
Description	Pointer to the base of array which stores the offset infomation of each IB Channel configured to Core<x>.	
Verification method	The structure member is generated as Spi_ChannelOffsets_Core<x> reference to base address of array which stores the offset information for IB channel configured to Core <x>.	
Example(s)	**Action**	**Generated output**
	Configure atleast 1 QSPI HW to Core 0.	`Spi_ChannelOffsets_Core0`
	Configure atleast 1 QSPI HW to Core 1.	`Spi_ChannelOffsets_Core1`

1.2.2.5 Member: QSPIHwConfigPtr[SPI_MAX_HW_UNIT]

Table 78 QSPIHwConfigPtr[SPI_MAX_HW_UNIT]

Name	QSPIHwConfigPtr[SPI_MAX_HW_UNIT]	
Type	Spi_QspiHwConfigType	
Description	Array of QSPI HW configuration.	
Verification method	If a core is allocated with atleast one QSPI<x> resource, The member is generated with array of base addresses of Hardware configuration structure &Spi_kQspiHwConfigQSPI<x> for each configured QSPI HW to core. If a core is not allocated with QSPI<x> resource, HW configuration is generated as NULL_PTR for a QSPI<x>.	
Example(s)	**Action**	**Generated output**

	Configure QSPI0 and QSPI1
	Allocate both QSPI0 and QSPI1 to CORE0
		`{`
		`&Spi_kQspiHwConfigQSPI0,`
		`&Spi_kQspiHwConfigQSPI1,`
		`NULL_PTR,`
		`NULL_PTR,`
		`NULL_PTR,`
		`NULL_PTR,`
		`}`

	Configure QSPI0
	Allocate QSPI0 to CORE0
	Configure QSPI1
	Allocate QSPI1 to CORE1
		Member in Spi_Config_Core0 looks like
		{
		&Spi_kQspiHwConfigQSPI0,
		&NULL_PTR,
		NULL_PTR,
		NULL_PTR,
		NULL_PTR,
		NULL_PTR,
		}
		Member in Spi_Config_Core1 looks like
		{
		NULL_PTR,
		&Spi_kQspiHwConfigQSPI1,
		NULL_PTR,
		NULL_PTR,
		NULL_PTR,
		NULL_PTR,
		}

1.2.2.6 Member: QSPIHwMap

Table 79 QSPIHwMap

Name	QSPIHwMap	
Type	Spi_QSPIHwMapConfigType	
Description	Indicates the communication mode for each QSPI if configured.	
Verification method	The member is shared by all QSPI HW. (bit2-bit0 for QSPI0, bit5-bit3 for QSPI1, bit8-bit6 for QSPI2, bit11-bit9 for QSPI3, bit14-bit12 for QSPI4, bit17-bit15 for QSPI5) The representation of 3 bits is as follows. (0x0 - QSPI is not configured for the core) (0x1 - QSPI is configured as synchronous for the core) (0x2 - QSPI is configured as Asynchronous for the core)	
Example(s)	**Action**	**Generated output**

	Configure QSPI0 for Asynchornous communication and Assign to Core0	`0x00002`
	Configure QSPI0 for synchornous communication and Assign to Core1	`0x00001`
	Configure QSPI0 for synchornous communication and Assign to Core0 Configure QSPI1 for Asynchornous communication and Assign to Core0	`0x00011`

1.2.2.7 Member: NoOfSequences

Table 80 NoOfSequences

Name	NoOfSequences	
Type	Spi_SequenceType	
Description	Number of sequences mapped to core.	
Verification method	The member is generated based on the number of sequences mapped to a core. Mapping of Sequences/jobs/channels to a core is based on the following methodology. • Identify the QSPI HW assigned to core. Resource allocation is done in ResourceMMcalConfig/ResourceMMcalConfig_0/ResourceMMcalCore. Example: QSPI0 and QSPI1 are assigned to Core0. • Identify the external devices which are using the QSPI HW assigned to core in above step. Example: SpiDriver/SpiExternalDevice/SpiExternalDevice_0 is using QSPI0 and SpiDriver/SpiExternalDevice /SpiExternalDevice_1 is using QSPI1. • Identify the Jobs which are driving these external devices. Example: SpiDriver/SpiJob/SpiJob_0 is driving SpiDriver/SpiExternalDevice SpiExternalDevice_0 and SpiDriver/SpiJob/SpiJob_1 is driving SpiDriver/SpiExternalDevice /SpiExternalDevice_1. SpiDriver/SpiJob /SpiJob_0 contain 4 channels and SpiDriver/SpiJob /SpiJob_1 contains 2 channels. • Identify the Jobs belong to a specific sequence using SpiSequence_0/SpiJobAssignment Example: SpiJob_0 belongs to SpiSequence_0 and SpiJob_1 belongs to Spi_Sequence_1 "So total number of sequences configured per core0 is '2'" "So total number of jobs configured per core0 is '2'" "So total number of channels configured per core0 is '6'"	
Example(s)	**Action**	**Generated output**
	Configure 10 sequences.	`10`

	all sequences are mapped to a single core (Example: Core0)	
	Configure 1 sequence. all sequences are mapped to a single core (Example: Core1)	1

1.2.2.8 Member: NoOfJobs

Table 81 NoOfJobs

Name	NoOfJobs	
Type	Spi_JobType	
Description	Number of jobs mapped to core.	
Verification method	The member is generated based on the number of jobs mapped to a core. Mapping of Sequences/jobs/channels to a core is based on the following methodology. • Identify the QSPI HW assigned to core. Resource allocation is done in ResourceMMcalConfig/ResourceMMcalConfig_0/ResourceMMcalCore. Example: QSPI0 and QSPI1 are assigned to Core0. • Identify the external devices which are using the QSPI HW assigned to core in above step. Example: SpiDriver/SpiExternalDevice /SpiExternalDevice_0 is using QSPI0 and SpiDriver/SpiExternalDevice /SpiExternalDevice_1 is using QSPI1. • Identify the Jobs which are driving these external devices. Example: SpiDriver/SpiJob/SpiJob_0 is driving SpiDriver/SpiExternalDevice / SpiExternalDevice_0 and SpiDriver/SpiJob /SpiJob_1 is driving SpiDriver/SpiExternalDevice /SpiExternalDevice_1. SpiDriver/SpiJob /SpiJob_0 contains 4 channels and Spi/SpiJob_1 contains 2 channels. • Identify the Jobs belong to a specific sequence using SpiSequence_0/SpiJobAssignment Example: SpiJob_0 belongs to SpiSequence_0 and SpiJob_1 belongs to Spi_Sequence_1 “So total number of sequences configured per core0 is ‘2’” “So total number of jobs configured per core0 is ‘2’” “So total number of channels configured per core0 is ‘6’”	
Example(s)	**Action**	**Generated output**
	Configure 10 jobs. all jobs are mapped to a single core (Example: Core0)	10
	Configure 1 job. all jobs are mapped to a single core (Example: Core1)	1

1.2.2.9 Member: NoOfChannels

Table 82 NoOfChannels

Name	NoOfChannels	
Type	Spi_ChannelType	
Description	Number of channels mapped to core.	
Verification method	The member is generated based on the number of channels mapped to a core. Mapping of Sequences/jobs/channels to a core is based on the following methodology. • Identify the QSPI HW assigned to core. Resource allocation is done in ResourceMMcalConfig/ResourceMMcalConfig_0/ResourceMMcalCore. Example: QSPI0 and QSPI1 are assigned to Core0. • Identify the external devices which are using the QSPI HW assigned to core in above step. Example: SpiDriver/SpiExternalDevice / SpiExternalDevice_0 is using QSPI0 and SpiDriver/SpiExternalDevice /SpiExternalDevice_1 is using QSPI1. • Identify the Jobs which are driving these external devices. Example: SpiDriver/SpiJob /SpiJob_0 is driving SpiDriver/SpiExternalDevice / SpiExternalDevice_0 and SpiDriver/SpiJob /SpiJob_1 is driving SpiDriver/SpiExternalDevice /SpiExternalDevice_1. SpiDriver/SpiJob /SpiJob_0 contains 4 channels and Spi/SpiJob_1 contains 2 channels. • Identify the Jobs belong to a specific sequence using SpiSequence_0/SpiJobAssignment Example: SpiJob_0 belongs to SpiSequence_0 and SpiJob_1 belongs to Spi_Sequence_1 “So total number of sequences configured per core0 is ‘2’” “So total number of jobs configured per core0 is ‘2’” “So total number of channels configured per core0 is ‘6’”	
Example(s)	**Action**	**Generated output**
	Configure 10 channels. all channels are mapped to a single core (Example: Core0)	`10`
	Configure 1 channel. all channels are mapped to a single core (Example: Core1)	`1`

1.2.3 Structure: Spi_kSequenceConfig_Core<x>

Table 83 Spi_kSequenceConfig_Core<x>

Name	Spi_kSequenceConfig_Core<x>	
Type	Spi_SequenceConfigType	
Description	Configuration structure of SPI driver for all Sequences belonging to Core <x> which will be referenced in core specific configuration structure. (x ranges from 0 to 5)	
Verification method	The generated file has this structure if atleast one sequence is assigned to Core <x>.	
Example(s)	**Action**	**Generated output**
	Configure 1 sequence for QSPI0	`static const Spi_SequenceConfigType`

	Allocate QSPI0 to Core0	`Spi_kSequenceConfig_Core0[] =`
		`{`
		` /* Asynchronous Sequence[s] on QSPI0 */`
		`/* Sequence:SpiSequence_0 */`
		` {`
		`   SpiConf_SpiSequence_SpiSequence_0,`
		`   /* Notification function */`
		`   &EEp_Test_Notification_0,`
		`   /* Job linked list */`
		
		`SpiSequence_0_JobLinkPtr_Physical,`
		`/* Seq linked list, with jobs shared */`
		`SpiSequence_0_SeqSharePtr,`
		`/* No. of jobs in Seq */`
		`  2U,`
		`/* Seq Interruptible or not */`
		`  SPI_SEQ_INT_TRUE,`
		`/* Hw Module Used (b000001)*/`
		`  0x01U,`
		`/* Sync sequence = 0x00 or Async sequence = 0x01*/`
		`  0x01U  (This Parameter is Applicable only for AUTOSAR 4.2.2)`
		` }`
		`}`

1.2.3.1 Member: SequenceId

Table 84 SequenceId

Name	SequenceId	
Type	Spi_SequenceType	
Description	Indicates the Sequence ID. *Note: Refer section 1.1.72 for more information.*	
Verification method	The member is generated based on the Symbolic name given for the sequence. SpiConf_SpiSequence_<SymbolicNameofSequence>	
Example(s)	**Action**	**Generated output**
	SpiDriver/SpiSequence = SpiSequence_0	`SpiConf_SpiSequence_SpiSequence_0`
	SpiDriver /SpiSequence =	`SpiConf_SpiSequence_EEP_TEST`

	EEP_TEST	

1.2.3.2 Member: SeqNotification

Table 85 SeqNotification

Name	SeqNotification	
Type	Spi_SeqEndNotification	
Description	Pointer to the callback functions configured by the user.	
Verification method	If SpiLevelDelivered = 1 or 2 and notification container(SpiSequence/ SpiSeqEndNotification) has a valid node, The structure member is generated with function name or address configured in the configuration parameter SpiSeqEndNotification if configured, otherwise the member is generated as NULL_PTR. If SpiLevelDelivered = 0, the member is not generated.	
Example(s)	**Action**	**Generated output**
	Configure SpiSeqEndNotification = 23245	`0x00005acd`
	Configure SpiSeqEndNotification = NULL_PTR.	`NULL_PTR`
	Don’t configure the SpiSeqEndNotification	`NULL_PTR`
	Configure SpiSeqEndNotification = EEp_Test_Notification_0	`&EEp_Test_Notification_0`

1.2.3.3 Member: JobLinkPtrPhysical

Table 86 JobLinkPtrPhysical

Name	JobLinkPtrPhysical	
Type	Spi_JobType*	
Description	Pointer to the base of array which stores the linked jobs for a sequence.	
Verification method	The structure member is generated as <Spi/SpiSequenceName>_ JobLinkPtr_Physical. *Note: <Spi/SpiSequenceName> represents the symbolic name given for a sequence.*	
Example(s)	**Action**	**Generated output**
	Configure a Sequence Spi/SpiSequence= SpiSequence_0	`SpiSequence_0_JobLinkPtr_Physical`
	Configure a Sequence Spi/SpiSequence= EEP_TEST	`EEP_TEST_JobLinkPtr_Physical`

1.2.3.4 Member: SeqSharePtr

Table 87 SeqSharePtr

Name	SeqSharePtr	
Type	uint8*	
Description	Pointer to the base of array which stores the sequence ID's with which the current sequence is sharing the jobs.	
Verification method	If SpiLevelDelivered configuration parameter is '0' then the member is not generated. Else the structure member is generated as <Spi/SpiSequenceName>_ SeqSharePtr . *Note: <Spi/SpiSequenceName> represents the symbolic name given for a sequence.*	
Example(s)	**Action**	**Generated output**
	Configure a Sequence Spi/SpiSequence= SpiSequence_0	`SpiSequence_0_SeqSharePtr`
	Configure a Sequence Spi/SpiSequence= EEP_TEST	`EEP_TEST_SeqSharePtr`

1.2.3.5 Member: NoOfJobInSeq

Table 88 NoOfJobInSeq

Name	NoOfJobInSeq	
Type	uint16	
Description	Indicates numbers of jobs assigned for a sequence.	
Verification method	The member is generated by counting the jobs listed under 'Spi/SpiSequence/SpiJobAssignment' for a sequence	
Example(s)	**Action**	**Generated output**
	Configure SpiSequence_0 with 10 jobs	`10`
	Configure SpiSequence_0 with 5 jobs	`5`

1.2.3.6 Member: SeqInterruptible

Table 89 SeqInterruptible

Name	SeqInterruptible
Type	uint8
Description	Indicates whether a sequence is interruptiable or non-interruptiable.
Verification method	If SpiLevelDelivered configuration parameter is '0' then the member is not generated. If SpiLevelDelivered configuration parameter is '1 or 2' and SpiInterruptibleSeqAllowed configuration parameter is 'false', then the member is not generated. Otherwise the member is generated as 'SPI_SEQ_INT_TRUE' if SpiInterruptibleSequence = true, else the member is generated as 'SPI_SEQ_INT_FALSE' if

	SpiInterruptibleSequence= false	
Example(s)	**Action**	**Generated output**
	SpiLevelDelivered = 0	`Member is not generated`

	SpiLevelDelivered = 1 or 2
	SpiInterruptibleSeqAllowed = false
		`Member is not generated`

	SpiLevelDelivered = 1 or 2
	SpiInterruptibleSeqAllowed = true
	SpiInterruptibleSequence = true
		`SPI_SEQ_INT_TRUE`

	SpiLevelDelivered = 1 or 2
	SpiInterruptibleSeqAllowed = true
	SpiInterruptibleSequence = false
		`SPI_SEQ_INT_FALSE`

1.2.3.7 Member: HwModuleUsed

Table 90 HwModuleUsed

Name	HwModuleUsed	
Type	uint8	
Description	Indicates which QSPI HW module is used for transmitting the sequence.	
Verification method	This member is shared by all QSPI HW kernels. Value 0 is used to represent QSPI0 Value 1 is used to represent QSPI1 Value 2 is used to represent QSPI2 Value 3 is used to represent QSPI3 Value 4 is used to represent QSPI4 Value 5 is used to represent QSPI5 If a sequence uses QSPI<x> for transmission then the macro is generated with a value where that particular bit is set.	
Example(s)	**Action**	**Generated output**
	SpiSequence_0 uses QSPI0 for transmission	`0x00`
	SpiSequence_0 uses QSPI1 for transmission	`0x01`
	SpiSequence_0 uses QSPI2 for transmission	`0x02`
	SpiSequence_0 uses QSPI3 for transmission	`0x03`
	SpiSequence_0 uses QSPI4 for transmission	`0x04`
	SpiSequence_0 uses QSPI5 for transmission	`0x05`

1.2.3.8 Member: u8Comm

Table 91 u8Comm

Name	u8Comm	
Type	uint8	
Description	Indicates whether a sequence is synchronous or asynchronous. Note: This Parameter is Applicable only for AUTOSAR 4.2.2.	
Verification method	The member is generated based on the Synchronous or asynchronous jobs assigned to the sequence. If all Synchronous jobs assigned to sequence then the member is generated as ‘0x0’ If all Asynchronous jobs assigned to sequence then the member is generated as ‘0x1’	
Example(s)	**Action**	**Generated output**
	SpiLevelDelivered = 0 Create a Sequence (SpiDriver/SpiSequence/SpiSequence_0) and assign with all synchronous jobs.	`0x0`
	SpiLevelDelivered = 1 Create a Sequence (SpiDriver/SpiSequence/SpiSequence_0) and assign with all Asynchronous jobs.	`0x1`
	SpiLevelDelivered = 2 Create a Sequence (SpiDriver/SpiSequence/SpiSequence_0) and assign with all synchronous jobs.	`0x0`
	SpiLevelDelivered = 2 Create a Sequence (SpiDriver/SpiSequence/SpiSequence_0) and assign with all Asynchronous jobs.	`0x1`

1.2.4 Structure: Spi_kJobConfig_Core<x>

Table 92 Spi_kJobConfig_Core<x>

Name	Spi_kJobConfig_Core<x>	
Type	Spi_JobConfigType	
Description	Configuration structure of SPI driver for all jobs belonging to Core <x> which will be referenced in core specific configuration structure. (x ranges from 0 to 5)	
Verification method	The generated file has this structure if atleast one Job is assigned to Core <x>.	
Example(s)	**Action**	**Generated output**

	Configure atleast 1 job for QSPI0
	Allocate QSPI0 to Core0
		`static const Spi_JobConfigType Spi_kJobConfig_Core0[] =`
		`{`
		`  /* Asynchronous Job[s] on QSPI0 */`
		`  /* Job:SpiJob_0 */`
		`  {`

```
    SpiConf_SpiJob_SpiJob_0,
    &Job_Notif_0, /* Notification
function */
    Spi_BaudRateAndClockParam(  /*
Baudrate = 2.0E7Hz */
     (0x00U), (0x00U),/* TQ , LoopBack*/
     (0x01U), (0x00U),/*  Q , A */
     (0x00U), (0x01U), /*  B , C */
     (0x01U), (0x00U),/*  CPH , CPOL */
     (0x00U)   /*  PAREN  */
    ),
    Spi_IdleLeadTrailParam(
     (1U), (1U), /* IPRE,IDLE:   IdleA/B
delay = 1.0E-7s */
     (1U), (1U), /* LPRE,LEAD:   Lead
delay    = 1.0E-7s */
     (1U), (1U),/* TPRE, TRAIL: Trail
delay   = 1.0E-7s */
     (1U)
    ),
    SpiJob_0_ChannelLinkPtr_Physical, /*
Channel linked list Physical*/
    SPI_CS_VIA_HW_OR_NONE,   /*
CS_VIA_HW */
    (uint8)1U, /* Job Priority : 0...3*/
    (uint8)STD_LOW,  /* CS polarity */
     /* Chnl[bit:7:4],QSPI[3:0] */
    (uint8)((SPI_QSPI_CHANNEL0 <<
4U)|SPI_QSPI0_INDEX),
    SPI_PARITY_UNUSED,/* Parity support
*/
    (0U)  /*Frame based CS is disabled*/
  }
}
```

1.2.4.1 Member: JobId

Table 93 JobId

Name	JobId
Type	Spi_JobType
Description	Indicates the Job ID.

	Note: Refer section 1.1.73 for more information.	
Verification method	The member is generated based on the Symbolic name given for the job. SpiConf_SpiJob_<SymbolicNameofJob>	
Example(s)	**Action**	**Generated output**
	Spi/SpiJob = SpiJob_0	`SpiConf_SpiJob_SpiJob_0`
	Spi/SpiJob = EEP_TEST	`SpiConf_SpiJob_EEP_TEST`

1.2.4.2 Member: JobNotification

Table 94 JobNotification

Name	JobNotification	
Type	Spi_JobEndNotification	
Description	Pointer to the callback functions configured by the user.	
Verification method	If SpiLevelDelivered = 1 or 2 and notification container(SpiJob/ SpiJobEndNotification) has a valid node, The structure member is generated with function name or address configured in the configuration parameter SpiJobEndNotification if configured, otherwise the member is generated as NULL_PTR. If SpiLevelDelivered = 0, the member is not generated.	
Example(s)	**Action**	**Generated output**
	Configure SpiJobEndNotification = 23245	`0x00005acd`
	Configure SpiJobEndNotification = NULL_PTR.	`NULL_PTR`
	Don’t configure the SpiJobEndNotification	`NULL_PTR`
	Configure SpiJobEndNotification = EEp_Test_Notification_0	`&EEp_Test_Notification_0`

1.2.4.3 Member: BaudRateAndClockParam

Table 95 BaudRateAndClockParam

Name	Spi_BaudRateAndClockParam
Type	uint32
Description	Indicates the baudrate and sampling parameter configuration for job transmission.
Verification method	The member is generated based on the evaluation of MACRO Spi_BaudRateAndClockParam(TQ,LB,Q,A,B,C,CPH,CPOL,PAREN). If configuration parameter SpiAutoCalcBaudParams = true then the inputs to the macro TQ,Q,A,B,C are calaculated using the configuration parameter SpiBaudrate, else the inputs ared derived from the configuration parameters as mentioned below.

	TQ- Value derived from configuration parameter SpiBaudrateParams/SpiBaudParamTQ Q- Value derived from configuration parameter SpiBaudrateParams/SpiBaudParamQ A- Value derived from configuration parameter SpiBaudrateParams/ SpiBaudParamA B- Value derived from configuration parameter SpiBaudrateParams/ SpiBaudParamB C- Value derived from configuration parameter SpiBaudrateParams/ SpiBaudParamC Other inputs are derived as follows irrespective of the state of SpiAutoCalcBaudParams. LB- Value is set to ‘1’ If configuration parameter SpiInternalLoopBackSupport = true, else the value is set to ‘0’ CPH- value is set to ‘0’ if configuration parameter SpiDataShiftEdge = TRAILING, else the value is set to ‘1’. CPOL- value is set to ‘0’ if configuration parameter SpiShiftClockIdleLevel = LOW, else the value is set to ‘1’. PAREN- value is set to ‘0’ if configuration parameter SpiParitySupport = UNUSED, else the value is set to ‘1’ The definition of the macro is as follows. #define Spi_BaudRateAndClockParam(TQ,LB,Q,A,B,C,CPH,CPOL,PAREN) ((uint32)(((uint32)TQ << 16U) | ((uint32)LB << 30U) | ((uint32)Q) | ((uint32)A << 6U) | ((uint32)B << 8U) | ((uint32)C << 10U) | ((uint32)CPH << 12U) | ((uint32)CPOL << 13U) | ((uint32)PAREN << 14U))) **Note:** Back calculate the spi baudrate using generated values of TQ, Q, A, B, C in the formula Spi BaudRate = QspiFreq div ((TQ + 1)*(Q + 1)*(A + 1 + B + C)) and confirm against requested baudrate.	
Example(s)	**Action**	**Generated output**

	SpiAutoCalcBaudParams = true
	SpiBaudrate = 640000
	SpiInternalLoopBackSupport = true
	SpiParitySupport = UNUSED
	SpiDataShiftEdge = TRAILING
	SpiShiftClockIdleLevel = LOW
		`Spi_BaudRateAndClockParam( /* Baudrate = 640000.0Hz */`
		`(0x18U), (0x01U),/* TQ , LoopBack*/`
		`(0x00U), (0x02U), /* Q , A */`
		`(0x01U), (0x01U), /* B , C */`
		`(0x00U), (0x00U), /* CPH , CPOL*/`
		`(0x00U) /* PAREN */`
		`)`

	SpiAutoCalcBaudParams = true
	SpiBaudrate = 640000
	SpiInternalLoopBackSupport = false
	SpiParitySupport = EVEN
	SpiDataShiftEdge = LEADING
	SpiShiftClockIdleLevel = HIGH
		`Spi_BaudRateAndClockParam( /* Baudrate = 640000.0Hz */`
		`(0x18U), (0x00U), /* TQ , LoopBack */`
		`(0x00U), (0x02U), /* Q , A */`
		`(0x01U), (0x01U), /* B , C */`
		`(0x01U), (0x01U), /* CPH , CPOL */`
		`(0x01U) /* PAREN */`

	)

SpiAutoCalcBaudParams = false
SpiBaudrateParams/SpiBaudParamTQ = 2
SpiBaudrateParams/SpiBaudParamQ = 10
SpiBaudrateParams/SpiBaudParamA = 1
SpiBaudrateParams/SpiBaudParamB = 0
SpiBaudrateParams/SpiBaudParamC = 1
SpiInternalLoopBackSupport = true
SpiParitySupport = UNUSED
SpiDataShiftEdge = TRAILING
SpiShiftClockIdleLevel = LOW
	`Spi_BaudRateAndClockParam(  /* Baudrate = 808080.808080808Hz */`
	`(0x02U), (0x01U), /* TQ , LoopBack */`
	`(0x0aU), (0x01U), /*  Q , A       */`
	`(0x00U), (0x01U), /*  B , C       */`
	`(0x00U), (0x00U), /*  CPH , CPOL */`
	`(0x00U)     /*  PAREN         */`
	`)`

SpiAutoCalcBaudParams = false
SpiBaudrateParams/SpiBaudParamTQ = 2
SpiBaudrateParams/SpiBaudParamQ = 10
SpiBaudrateParams/SpiBaudParamA = 1
SpiBaudrateParams/SpiBaudParamB = 0
SpiBaudrateParams/SpiBaudParamC = 1
SpiInternalLoopBackSupport = true
SpiParitySupport = UNUSED
SpiDataShiftEdge = TRAILING
SpiShiftClockIdleLevel = LOW
	`Spi_BaudRateAndClockParam(  /* Baudrate = 808080.808080808Hz */`
	`(0x02U), (0x00U), /* TQ , LoopBack */`
	`(0x0aU), (0x01U), /*  Q , A       */`
	`(0x00U), (0x01U), /*  B , C       */`
	`(0x01U), (0x01U), /*  CPH , CPOL */`
	`(0x01U) /*  PAREN         */`
	`)`

1.2.4.4 Member: IdleLeadTrailDelay

Table 96 IdleLeadTrailDelay

Name	IdleLeadTrailDelay
Type	uint32
Description	Indicates the frame delay configuration for job transmission.
Verification method	The member is generated based on the evaluation of MACRO Spi_IdleLeadTrailParam (IPRE, IDLE, LPRE, LEAD, TPRE, TRAIL, PARTYP). The inputs (IPRE, IDLE, LPRE, LEAD, TPRE,TRAIL) to the macro vary based on the following conditions. **Condition 1:** If configuration parameter SpiAutoCalcDelayParams = true then the inputs to the macro are derived as follows. ▪ IPRE and IDLE are derived using the configuration parameter SpiIdleTime. ▪ LPRE and LEAD are derived using the configuration parameter SpiTimeClk2Cs. ▪ TPRE and TRAILare derived using the configuration parameter

	SpiTrailingTime. **Condition 2:** if Configuration parameter SpiAutoCalcDelayParams = false then the inputs to the macro are as follows. ▪ IPRE and IDLE are derived using the configuration parameter SpiDelayParamIdlePre, SpiDelayParamIdleLength repectively. ▪ LPRE and LEAD are derived using the configuration parameter SpiDelayParamLeadPre, SpiDelayParamLeadLength respectively. ▪ TPRE and TRAILare derived using the configuration parameter SpiDelayParamTrailPre, SpiDelayParamTrailLength respectively. PARTYP: input is set to ‘0’ if configuration parameter SpiParitySupport =ODD else, input is set to ‘1’ #define Spi_IdleLeadTrailParam (IPRE, IDLE, LPRE, LEAD, TPRE, TRAIL, PARTYP) ((uint32)(((uint32) IPRE << 1U) | ((uint32) IDLE << 4U) | ((uint32) LPRE << 7U) | ((uint32) LEAD << 10U) | ((uint32) TPRE << 13U) | ((uint32) TRAIL << 16U) | ((uint32) PARTYP << 19U)))	
Example(s)	**Action**	**Generated output**

	When Condition 1 is satisfied
	SpiAutoCalcDelayParams = true
	SpiIdleTime = 4.0E-8
	SpiTrailingTime = 4.0E-8
	SpiTimeClk2Cs=1.0E-7
	SpiParitySupport = ODD
		`Spi_IdleLeadTrailParam(`
		`(0U), (2U), /* IPRE,IDLE: IdleA/B delay = 4.0E-8s */`
		`(1U), (1U), /* LPRE,LEAD: Lead delay = 1.0E-7s */`
		`(0U), (2U),/* TPRE, TRAIL: Trail delay = 4.0E-8s */`
		`(0U)`
		`)`

	When Condition 2 is satisfied
	SpiAutoCalcDelayParams = false
	SpiIdleTime = 4.0E-8
	SpiTrailingTime = 4.0E-8
	SpiTimeClk2Cs=1.0E-7
	SpiParitySupport = EVEN
	SpiDelayParamTrailPre = 0
	SpiDelayParamTrailLength = 7
	SpiDelayParamIdlePre = 0
	SpiDelayParamIdleLength = 1
	SpiDelayParamLeadPre= 0
	SpiDelayParamLeadLength=7
		`Spi_IdleLeadTrailParam(`
		`(0U), (1U), /* IPRE,IDLE: IdleA/B delay = 1.0E-7s */`
		`(0U), (7U), /* LPRE,LEAD: Lead delay = 1.0E-7s */`
		`(0U), (7U),/* TPRE, TRAIL: Trail delay = 1.0E-7s */`
		`(1U)`
		`)`

1.2.4.5 Member: ChnlLinkPtrPhysical

Table 97 ChnlLinkPtrPhysical

Name	ChnlLinkPtrPhysical	
Type	Spi_ChannelType*	
Description	Pointer to the base of array which stores the linked channels for a job.	
Verification method	The structure member is generated as <Spi/SpiJobName>_ ChannelLinkPtr_Physical. *Note: <Spi/ SpiJobName > represents the symbolic name given for a job.*	
Example(s)	**Action**	**Generated output**
	Configure a Job Spi/SpiJob= SpiJob_0	`SpiJob_0_ChannelLinkPtr_Physical`
	Configure a Job Spi/SpiJob= EEP_TEST	`EEP_TEST_ChannelLinkPtr_Physical`

1.2.4.6 Member: CSPortPin

Table 98 CSPortPin

Name	CSPortPin	
Type	uint16	
Description	Represents Chip Select PortPin and port information.	
Verification method	If configuration parameter SpiEnableCs = true and SpiCsSelection = CS_VIA_PERIPHERAL_ENGINE then the member is generated as 'SPI_CS_VIA_HW_OR_NONE . If configuration parameter SpiEnableCs = true and SpiCsSelection = CS_VIA_GPIO then the member is generated as ((SpiCsGpio/ SpiCsGpioPortSelection << 4) | (SpiCsGpio/SpiCsGpioPinSelection)) Bit representation of member is as follows. [bit3-bit0]: Specifies Port pin information. Range: [0..0xF] [bit15-bit4]: Specifies Port information. Range: [0..0xFFF] If configuration parameter SpiEnableCs = false then the member is generated as 'SPI_CS_VIA_HW_OR_NONE'	
Example(s)	**Action**	**Generated output**
	SpiEnableCs = false	`SPI_CS_VIA_HW_OR_NONE`

	SpiEnableCs = true
	SpiCsSelection = CS_VIA_GPIO
	SpiCsGpioPortSelection = 2
	SpiCsGpioPinSelection = 1
		`((2U << 4U)|(1U)), /* CS_VIA_GPIO */`

	SpiEnableCs = true
	SpiCsSelection = SPI_CS_VIA_HW_OR_NONE'
		`SPI_CS_VIA_HW_OR_NONE`

1.2.4.7 Member: JobPriority

Table 99 JobPriority

Name	JobPriority	
Type	uint8	
Description	Represents Job Priority ranging from 0 (Lowest) to 3 (Highest)	
Verification method	The member is generated based on the value assigned to the configuration parameter SpiJobPriority.	
Example(s)	**Action**	**Generated output**
	SpiJob/SpiJob_0/SpiJobPriority = 0	`0`
	SpiJob/SpiJob_0/SpiJobPriority = 1	`1`
	SpiJob/SpiJob_0/SpiJobPriority = 3	`3`

1.2.4.8 Member: CsPolarity

Table 100 CsPolarity

Name	CsPolarity	
Type	uint8	
Description	Represents the Chip select polarity	
Verification method	The member is generated based on the value assigned to the configuration parameter SpiCsPolarity.	
Example(s)	**Action**	**Generated output**
	SpiExternalDevice_0/ SpiCsPolarity = LOW	`STD_LOW`
	SpiExternalDevice_0/ SpiCsPolarity = HIGH	`STD_HIGH`

1.2.4.9 Member: HwUnit

Table 101 HwUnit

Name	HwUnit
Type	Spi_HWUnitType
Description	Represents the QSPI module and Hw Channel information
Verification method	The member is generated using the configuration parameters SpiCsIdentifier and SpiHwUnit as (SPI_QSPI_[< SpiCsIdentifier>] <<4 | SPI_[< SpiHwUnit>]_INDEX Bit representation of member is as follows. Range: [0..5] 0: QSPI0 1: QSPI1 2: QSPI2 3:QSPI3 4:QSPI4

	5:QSPI5 [bit7 - bit4]: Specifies the QSPI HW channel used. Range: [0..0xF]	
Example(s)	**Action**	**Generated output**

	SpiCsIdentifier = CHANNEL0
	SpiHwUnit = QSPI0
		`((SPI_QSPI_CHANNEL0 << 4U)|SPI_QSPI0_INDEX)`

	SpiCsIdentifier = CHANNEL1
	SpiHwUnit = QSPI0
		`((SPI_QSPI_CHANNEL1 << 4U)|SPI_QSPI0_INDEX)`

	SpiCsIdentifier = CHANNEL1
	SpiHwUnit = QSPI1
		`((SPI_QSPI_CHANNEL1 << 4U)|SPI_QSPI1_INDEX)`

	SpiCsIdentifier = CHANNEL15
	SpiHwUnit = QSPI5
		`((SPI_QSPI_CHANNEL15 << 4U)|SPI_QSPI5_INDEX)`

1.2.4.10 Member: ParitySupport

Table 102 JobPriority

Name	ParitySupport	
Type	uint8	
Description	Indicates the Parity used.	
Verification method	The member is generated based on the value assigned to the configuration parameter SpiParitySupport as SPI_PARITY_[< SpiParitySupport >]	
Example(s)	**Action**	**Generated output**
	SpiParitySupport = EVEN	`SPI_PARITY_EVEN`
	SpiParitySupport = ODD	`SPI_PARITY_ODD`
	SpiParitySupport = UNUSED	`SPI_PARITY_UNUSED`

1.2.4.11 Member: FramebasedCs

Table 103 FramebasedCs

Name	FramebasedCs	
Type	uint8	
Description	Indicates whether to toggle the CS after every frame transmission or not.	
Verification method	The member is generated as ‘1’ if SpiFrameBasedCS = true else the member is set to ‘0’	
Example(s)	**Action**	**Generated output**
	SpiFrameBasedCS = true	`1`
	SpiFrameBasedCS = false	`0`

1.2.5 Structure: Spi_kChannelConfig_Core<x>

Table 104 Spi_kChannelConfig_Core<x>

Name	Spi_kChannelConfig_Core<x>	
Type	Spi_ChannelConfigType	
Description	Configuration structure of SPI driver for all channels belonging to Core <x> which will be referenced in core specific configuration structure. (x ranges from 0 to 5)	
Verification method	The generated file has this structure if atleast one channel is assigned to Core <x>.	
Example(s)	**Action**	**Generated output**

	Configure atleast 1 channel for QSPI0
	Allocate QSPI0 to Core0
		static const Spi_ChannelConfigType Spi_kChannelConfig_Core0[] =
		{
		    /* Channel:SpiChannel_0 */
		  {
		    0x00000000U, /* Default data */
		    0x1ffeU, /* Number of Data Elements */
		    SPI_EB_CHANNEL, /* External Buffer Channel */
		    0x08U,/* LSB[7], DataWidth=8[6:0] */
		    SpiConf_SpiChannel_SpiChannel_0
		  }
		}

1.2.5.1 Member: Defaultdata

Table 105 Defaultdata

Name	Defaultdata	
Type	uint32	
Description	Indicates the default data to be transmitted.	
Verification method	If SpiDefaultData is configured, the member is generated based on the value assigned to SpiDefaultData , else the member is generated as '0'	
Example(s)	**Action**	**Generated output**
	SpiDefaultData = 255	0x000000ff
	SpiDefaultData = 0	0x00000000
	If SpiDefaultData is not configured	0x00000000

1.2.5.2 Member: NoOfDataElements

Table 106 NoOfDataElements

Name	NoOfDataElements
Type	uint16
Description	In case of IB, Indicates the number of Data elements to be transmitted.

	In case of EB, indicates the maximum numbers of data elements are allowed.	
Verification method	If SpiChannelType = EB, the member is generated based on the value assigned to SpiEbMaxLength. If SpiChannelType = IB, the member is generated based on the value assigned to SpiIbNBuffers.	
Example(s)	**Action**	**Generated output**

	SpiChannelType = EB
	SpiEbMaxLength = 8190
		`0x1ffe`

	SpiChannelType = IB
	SpiIbNBuffers = 10
		`0x000a`

1.2.5.3 Member: ChannelType

Table 107 ChannelType

Name	ChannelType	
Type	uint8	
Description	Indicates EB or IB channel.	
Verification method	If SpiChannelType = EB, the member is generated based as SPI_EB_CHANNEL. If SpiChannelType = IB, the member is generated based as SPI_IB_CHANNEL.	
Example(s)	**Action**	**Generated output**
	SpiChannelType = EB	`SPI_EB_CHANNEL`
	SpiChannelType = IB	`SPI_IB_CHANNEL`

1.2.5.4 Member: QSPIHwUnit

Table 108 QSPIHwUnit

Name	QSPIHwUnit	
Type	uint8	
Description	Lower nibble incidates the QSPI HW unit information and upper nibble indicates type of communication (synchronous and Asynchronous).	
Verification method	If QSPI HW is configured for Aysnchronous communicaton, the member is generated as (0x1 << 4 | QSPI_INDEX) QSPI_INDEX ranges from 0..5 If QSPI is configured for Synchronous communication, the member is generated as ‘0’. If SpiChannelBuffersAllowed is ‘1’, then the member is not generated.	
Example(s)	**Action**	**Generated output**
	SpiChannelBuffersAllowed =1	`Member is not generated.`

	SpiChannelBuffersAllowed =0
	SpiHwUnit = 0
	SpiHwUnitSynchronous= ASYNCHRONOUS
		`0x10`

	SpiChannelBuffersAllowed =0
	SpiHwUnit = 1
		`0x00`

	SpiHwUnitSynchronous= SYNCHRONOUS	

	SpiChannelBuffersAllowed =2
	SpiHwUnit = 1
	SpiHwUnitSynchronous= SYNCHRONOUS
		`0x00`

1.2.5.5 Member: DataConfig

Table 109 DataConfig

Name	DataConfig	
Type	uint8	
Description	Indicates the Data characteristics like Data width and Trasfer start (MSB/LSB) first.	
Verification method	The member is generated using configuration parameters SpiDataWidth and SpiTransferStart. Bit representation of member is as follows. Bit7: set to '1' if SpiTransferStart = MSB, else set to '0' Bit6-bit0: set to the value assigned for SpiDataWidth.	
Example(s)	**Action**	**Generated output**

	SpiTransferStart = LSB
	SpiDataWidth = 8
		`0x08`

	SpiTransferStart = LSB
	SpiDataWidth = 16
		`0x10`

	SpiTransferStart = MSB
	SpiDataWidth = 32
		`0xA0`

1.2.5.6 Member: ChannelId

Table 110 ChannelId

Name	ChannelId	
Type	Spi_ChannelType	
Description	Indicates the Channel ID. *Note: Refer section 1.1.74 for more information.*	
Verification method	The member is generated based on the Symbolic name given for the channel. SpiConf_SpiChannel_<SymbolicNameofChannel>	
Example(s)	**Action**	**Generated output**
	Spi/SpiChannel = SpiChannel_0	`SpiConf_SpiChannel_SpiChannel_0`
	Spi/SpiChannel = EEP_TEST	`SpiConf_SpiChannel_EEP_TEST`

1.2.6 Structure: Spi_kQspiHwConfigQSPI<x>

Table 111 Spi_kQspiHwConfigQSPI<x>

Name	Spi_kQspiHwConfigQSPI<x>	
Type	Spi_QspiHwConfigType	
Description	HW Configuration structure for QSPI<x> (x ranges from 0 to 5).	
Verification method	The generated file has this structure if atleast one QSPI HW is configured.	
Example(s)	**Action**	**Generated output**

	Configure atleast 1 QSPI and assign to Core0	`static const Spi_QspiHwConfigType Spi_kQspiHwConfigQSPI1 =`
		`{`
		`  0x00040000U, /* Active CS Level, SSOC SFR value */`
		`  SPI_JOB_QUEUE_LENGTH_QSPI1,/* Job Queue Length */`
		`  (uint8)0U, /* DMA Rx Channel */`
		`  (uint8)1U, /* DMA Tx Channel */`
		`  SPI_DMA_MAX_TCS_NUM_QSPI1, /* DMA TCS count, for both Rx and Tx */`
		`  SPI_CLK_SLEEP_DISABLE,     /* Module Sleep disabled */`
		`  (uint8)1U,  /* Input class, MRIS bit field in PISEL SFR */`
		`  1U,  /* Max Sequence Count on the QSPI */`
		`  0U,  /* External Demultiplexer feature is disabled */`
		`  0U /* SLSO0 Strobe delay */`
		`};`

1.2.6.1 Member: ActiveChipSelectLevel

Table 112 ActiveChipSelectLevel

Name	ActiveChipSelectLevel	
Type	uint32	
Description	Indicates the Chip select configuration.	
Verification method	The member is generated based on the configuration parameters SpiCsIdentifier and SpiCsPolarity if SpiEnableCs = true and SpiCsSelection = CS_VIA_PERIPHERAL_ENGINE. Otherwise the member is set to ‘0’. **calculation method:** In ActiveChipSelectLevel value, the upper 16-bits represents the SpiCsIdentifier and lower 16-bits represents the SpiCsPolarity of respective SpiCsIdentifier channel.	
Example(s)	**Action**	**Generated output**

	SpiCsIdentifier = CHANNEL5
	SpiCsPolarity = LOW
	[Upper 16-bits = 0000000000100000 in hex 0020
		`0x00200000`

	Lower 16-bits = 0000000000000000 in hex 0000]	

	SpiCsIdentifier = CHANNEL15
	SpiCsPolarity = LOW
	[Upper 16-bits = 1000000000000000 in hex 8000
	Lower 16-bits = 0000000000000000 in hex 0000]
		`0x80000000`

	SpiCsIdentifier = CHANNEL0
	SpiCsPolarity = LOW
	[Upper 16-bits = 0000000000000001 in hex 0001
	Lower 16-bits = 0000000000000000 in hex 0000]
		`0x00010000`

	SpiCsIdentifier = CHANNEL0
	SpiCsPolarity = HIGH
	[Upper 16-bits = 0000000000000001 in hex 0001
	Lower 16-bits = 0000000000000001 in hex 0001]
		`0x00010001`

1.2.6.2 Member: JobQueueLength

Table 113 JobQueueLength

Name	JobQueueLength	
Type	uint16	
Description	Specifies the length for the job queue for the corresponding QSPI HW.	
Verification method	If configuration parameter SpiHwUnitSynchronous is set to 'ASYNCHRONOUS' then macro is generated as SPI_JOB_QUEUE_LENGTH_QSPI<x>(x ranges from 0..5) based on the QSPI HW used SpiHwUnit. If SpiHwUnitSynchronous = SYNCHRONOUS or SpiLevelDelivered = 0, the member is set to '0'.	
Example(s)	**Action**	**Generated output**
	SpiLevelDelivered = 0	`0`

	SpiLevelDelivered = 1
	SpiHwUnit = QSPI1
	Jobs are configured
		`SPI_JOB_QUEUE_LENGTH_QSPI1`

	SpiLevelDelivered = 1
	SpiHwUnit = QSPI0
	Jobs are configured
		`SPI_JOB_QUEUE_LENGTH_QSPI0`

	SpiLevelDelivered = 2
	SpiHwUnitSynchronous = ASYNCHRONOUS
	SpiHwUnit = QSPI2
	Jobs are configured
		`SPI_JOB_QUEUE_LENGTH_QSPI2`

	SpiLevelDelivered = 2
	SpiHwUnitSynchronous = SYNCHRONOUS
	SpiHwUnit = QSPI0
	Jobs are configured
		`0`

1.2.6.3 Member: DMARxChannel

Table 114 DMARxChannel

Name	DMARxChannel	
Type	uint8	
Description	Indicates the DMA channels used for receive.	
Verification method	If configuration parameter SpiLevelDelivered is set to 1 or 2 and SpiHwUnitSynchronous = ASYNCHRONOUS, the member is generated based on the DMA channel configured : ref(SpiHwDmaChannelReceptionRef)/ DmaChannelId. If configuration parameter SpiLevelDelivered is set to 2 and SpiHwUnitSynchronous = SYNCHRONOUS, the member is generated as SPI_DMA_CHNL_INVALID. If SpiLevelDelivered = 0, the member is not generated.	
Example(s)	**Action**	**Generated output**
	SpiLevelDelivered = 0	`Member is not generated`
	SpiLevelDelivered = 1 SpiHwDmaChannelReceptionRef /DmaChannelId = 1	`1`
	SpiLevelDelivered = 1 SpiHwDmaChannelReceptionRef /DmaChannelId = 3	`3`

	SpiLevelDelivered = 2
	SpiHwUnitSynchronous = ASYNCHRONOUS
	SpiHwDmaChannelReceptionRef /DmaChannelId = 4
		`4`

	SpiLevelDelivered = 2
	SpiHwUnitSynchronous = SYNCHRONOUS
		`SPI_DMA_CHNL_INVALID`

1.2.6.4 Member: DMATxChannel

Table 115 DMATxChannel

Name	DMATxChannel	
Type	uint8	
Description	Indicates the DMA channels used for transmission.	
Verification method	If configuration parameter SpiLevelDelivered is set to 1 or 2 and SpiHwUnitSynchronous = ASYNCHRONOUS, the member is generated based on the DMA channel configured: ref (SpiHwDmaChannelTransmissionRef)/ DmaChannelId. If configuration parameter SpiLevelDelivered is set to 2 and SpiHwUnitSynchronous = SYNCHRONOUS, the member is generated as SPI_DMA_CHNL_INVALID. If SpiLevelDelivered = 0, the member is not generated.	
Example(s)	**Action**	**Generated output**
	SpiLevelDelivered = 0	`Member is not generated`
	SpiLevelDelivered = 1 SpiHwDmaChannelTransmissionRef /DmaChannelId = 1	`1`
	SpiLevelDelivered = 1 SpiHwDmaChannelTransmissionRef	`3`

	/DmaChannelId = 3	

	SpiLevelDelivered = 2
	SpiHwUnitSynchronous = ASYNCHRONOUS
	SpiHwDmaChannelTransmissionRef /DmaChannelId = 4
		`4`

	SpiLevelDelivered = 2
	SpiHwUnitSynchronous = SYNCHRONOUS
		`SPI_DMA_CHNL_INVALID`

1.2.6.5 Member: DMATCSCount

Table 116 DMATCSCount

Name	DMATCSCount	
Type	uint8	
Description	Indicates the number of channel assigned to an Asychronous job.	
Verification method	If configuration parameter SpiLevelDelivered is set to 1 or 2 and SpiHwUnitSynchronous = ASYNCHRONOUS, the member is generated as SPI_DMA_MAX_TCS_NUM_QSPI<x>. If configuration parameter SpiLevelDelivered is set to 2 and SpiHwUnitSynchronous = SYNCHRONOUS, the member is set to 0. If SpiLevelDelivered = 0, the member is not generated.	
Example(s)	**Action**	**Generated output**
	SpiLevelDelivered = 0	`Member is not generated`
	SpiLevelDelivered = 1 Configure QSPI0 Configure the channels and assign to Job	`SPI_DMA_MAX_TCS_NUM_QSPI0`
	SpiLevelDelivered = 1 Configure QSPI1 Configure the channels and assign to Job	`SPI_DMA_MAX_TCS_NUM_QSPI1`

	SpiLevelDelivered = 2
	SpiHwUnitSynchronous = ASYNCHRONOUS
	Configure QSPI2
	Configure the channels and assign to Job
		`SPI_DMA_MAX_TCS_NUM_QSPI2`

	SpiLevelDelivered = 2
	SpiHwUnitSynchronous = SYNCHRONOUS
		`0`

1.2.6.6 Member: ClockSetting

Table 117 ClockSetting

Name	ClockSetting
Type	uint8

Description	Enables/disables the sleep.	
Verification method	The member is generated as 'SPI_CLK_SLEEP_DISABLE' if SpiHwConfigurationQspi/SpiSleepEnableQspix is set to false, else the member is generated as SPI_CLK_SLEEP_ENABLE	
Example(s)	**Action**	**Generated output**
	SpiSleepEnableQspix = false	`SPI_CLK_SLEEP_DISABLE`
	SpiSleepEnableQspix = true	`SPI_CLK_SLEEP_ENABLE`

1.2.6.7 Member: MasterReceivePortPin

Table 118 MasterReceivePortPin

Name	MasterReceivePortPin	
Type	uint8	
Description	Indicates the Master receive pin used.	
Verification method	The member is generated based on the configuration parameter SpiHWPinMRSTQspix. If the value assigned for SpiHWPinMRSTQspix contains text 'A_, B_,C_,CN_,CP_,D_,DN_,DP_,E_,F_,FN_,FP_,G_,H_' then the member is generated 0,1,2,2,2,3,3,3,4,5,5,5,6,7 respectively.	
Example(s)	**Action**	**Generated output**
	SpiHWPinMRSTQspix = MRST1B_PORT11_PIN3 Configure QSPI1	`1`
	SpiHWPinMRSTQspix = MRST1A_PORT10_PIN1 Configure QSPI1	`0`

1.2.6.8 Member: MaxSequence

Table 119 MaxSequence

Name	MaxSequence	
Type	Spi_SequenceType	
Description	Total number of sequences configured per QSPI HW	
Verification method	The value for the member is generated by counting all the configured sequences under Spi/SpiSequence per QSPI HW	
Example(s)	**Action**	**Generated output**
	Configure 10 sequences for QSPI0	`10`
	Configure 5 sequences for QSPI0	`5`

1.2.6.9 Member: ExternalDemuxEnabled

Table 120 ExternalDemuxEnabled

Name	ExternalDemuxEnabled

Type	uint8	
Description	Indicates the external demultiplexer feature is enabled or disabled for a QSPI HW.	
Verification method	The value for the member is generated based on the value assigned to configuration parameter SpiExternalDemux. **Note:** The configuration parameter SpiExternalDemux is allowed to configure only when the property file variable Spi.QSPIxExternalDemux is ON. (x ranges from 0-5)	
Example(s)	**Action**	**Generated output**
	For QSPI0 Spi.QSPI0ExternalDemux:OFF	`0`
	For QSPI0 Spi.QSPI0ExternalDemux:ON SpiExternalDemux = true	`1`
	For QSPI0 Spi.QSPI0ExternalDemux:ON SpiExternalDemux = false	`0`

1.2.6.10 Member: StrobeDelay

Table 121 StrobeDelay

Name	StrobeDelay	
Type	uint32	
Description	Indicates the strobe delay to be configured for external demultiplexer feature.	
Verification method	The value for the member is generated based on the value assigned to configuration parameter SpiSLSO0StrobeDelay. **Note:** The configuration parameter SpiSLSO0StrobeDelay is allowed to configure only when the configuration parameter SpiExternalDemux = true and Spi.QSPIxExternalDemux is ON. (x ranges from 0-5)	
Example(s)	**Action**	**Generated output**
	For QSPI0 Spi.QSPI0ExternalDemux:OFF	`0`

	For QSPI0
	Spi.QSPI0ExternalDemux:ON
	SpiExternalDemux = true
	SpiSLSO0StrobeDelay = 2
		`0x00000002`

	For QSPI0
	Spi.QSPI0ExternalDemux:ON
	SpiExternalDemux = true
	SpiSLSO0StrobeDelay = 31
		`0x0000001f`

	For QSPI0
	Spi.QSPI0ExternalDemux:ON
	SpiExternalDemux = true
	SpiSLSO0StrobeDelay = 15
		`0x0000000f`

	For QSPI0 Spi.QSPI0ExternalDemux:ON SpiExternalDemux = false	`0`

1.2.7 Structure: Spi_ChannelOffsets_Core<x>

Table 122 Spi_ChannelOffsets_Core<x>

<table>
<tr><td><b>Name</b></td><td colspan="2">Spi_ChannelOffsets_Core<x></td></tr>
<tr><td><b>Type</b></td><td colspan="2">Spi_CoreChannelOffsetType</td></tr>
<tr><td><b>Description</b></td><td colspan="2">Array of structures holding the offset value and number of data elements required for each IB channel in core<x> buffer.</td></tr>
<tr><td><b>Verification method</b></td><td colspan="2">Each array element is a structure which holds the offset for the channel and number of data elements to be transmitted for the channel.
The last element of the array holds 0xFFFF for offset and 0xFFFF for number of elements. This last element is just used to mark the end.
For EB channels, the respective array element structure members are set to ‘0’.
<i>Note: For every channel, the algorithm ensures that the channel buffer allocated is word aligned.</i></td></tr>
<tr><td rowspan="3"><b>Example(s)</b></td><td><b>Action</b></td><td><b>Generated output</b></td></tr>
<tr><td>Allocate QSPI0 to Core0
Configure 3 IB channels
SpiChannel_0/SpiDataWidth = 8
SpiChannel_0/ SpiIbNBuffers = 10
SpiChannel_1/SpiDataWidth = 16
SpiChannel_1/ SpiIbNBuffers = 10
SpiChannel_2/SpiDataWidth = 32
SpiChannel_2/ SpiIbNBuffers = 10</td><td><code>static const Spi_CoreChannelOffsetType Spi_ChannelOffsets_Core0</code>
<code>[SPI_NUM_IB_CHANNELS_CORE0 + SPI_NUM_EB_CHANNELS_CORE0 + 1U] =</code>
<code>{</code>
<code> {0, 10}, /* for IB channels SpiChannel_2 */</code>
<code> {40, 10}, /* for IB channels SpiChannel_1 */</code>
<code> {60, 10}, /* for IB channels SpiChannel_0 */</code>
<code> {0xFFFF, 0xFFFF}</code>
<code>};</code></td></tr>
<tr><td>Configure 3 channels, and mark all the channel types to EB
SpiChannelType= EB for all channels.
Configure Core0 with QSPI0</td><td><code>static const Spi_CoreChannelOffsetType Spi_ChannelOffsets_Core0</code>
<code>[SPI_NUM_IB_CHANNELS_CORE0 + SPI_NUM_EB_CHANNELS_CORE0 + 1U] =</code>
<code>{</code>
<code> {0, 0},   /*SpiChannel_2*/</code>
<code> {0, 0},   /*SpiChannel_1*/</code>
<code> {0, 0},   /*SpiChannel_0*/</code>
<code> {0xFFFF, 0xFFFF}</code>
<code>};</code></td></tr>
</table>

	Allocate QSPI0 to Core0
	Configure 3 channels
	SpiChannel_0/SpiDataWidth = 8
	SpiChannel_0/SpiChannelType = IB
	SpiChannel_0/ SpiIbNBuffers = 10
	SpiChannel_1/SpiDataWidth = 16
	SpiChannel_1/SpiChannelType = EB
	SpiChannel_1/ SpiIbNBuffers = 10
	SpiChannel_2/SpiDataWidth = 32
	SpiChannel_2/ SpiIbNBuffers = 10
	SpiChannel_2/SpiChannelType = IB
		static const Spi_CoreChannelOffsetType Spi_ChannelOffsets_Core0
		[SPI_NUM_IB_CHANNELS_CORE0 + SPI_NUM_EB_CHANNELS_CORE0 + 1U] =
		{
		{0, 10}, /* for IB channels SpiChannel_2 */
		{40, 10}, /* for IB channels SpiChannel_0 */
		{0, 0}, /*SpiChannel_1*/
		{0xFFFF, 0xFFFF}
		};

	Allocate QSPI0 to Core0
	Configure 3 channels
	SpiChannel_0/SpiDataWidth = 8
	SpiChannel_0/SpiChannelType = IB
	SpiChannel_0/ SpiIbNBuffers = 10
	SpiChannel_1/SpiDataWidth = 8
	SpiChannel_1/SpiChannelType = IB
	SpiChannel_1/ SpiIbNBuffers = 10
	SpiChannel_2/SpiDataWidth = 8
	SpiChannel_2/ SpiIbNBuffers = 10
	SpiChannel_2/SpiChannelType = IB
		static const Spi_CoreChannelOffsetType Spi_ChannelOffsets_Core0
		[SPI_NUM_IB_CHANNELS_CORE0 + SPI_NUM_EB_CHANNELS_CORE0 + 1U] =
		{
		{0, 10}, /* for IB channels SpiChannel_2 */
		{12, 10}, /* for IB channels SpiChannel_1 */
		{24, 10}, /* for IB channels SpiChannel_0 */
		{0xFFFF, 0xFFFF}
		};

1.2.7.1 Member: ChannelOffset

Table 123 ChannelOffset

Name	ChannelOffset

Type	uint16	
Description	Indicates the channel offset for a channel in the buffer allocated for Core.	
Verification method	The member is generated as 0, if the channel SpiChannelType = EB The member is generated as valid offset(Word aligned) when SpiChannelType = IB	
Example(s)	**Action**	**Generated output**

	Configure 3 IB channels
	SpiChannel_0/SpiDataWidth = 8
	SpiChannel_0/ SpiIbNBuffers = 10
	SpiChannel_1/SpiDataWidth = 16
	SpiChannel_1/ SpiIbNBuffers = 10
	SpiChannel_2/SpiDataWidth = 32
	SpiChannel_2/ SpiIbNBuffers = 10
		`{0, 10},  /* for IB channels SpiChannel_2 */`
		`{40, 10},  /* for IB channels SpiChannel_1 */`
		`{60, 10},  /* for IB channels SpiChannel_0 */`

	Configure 3 channels, and mark all the channel types to EB
	SpiChannelType= EB for all channels.
	Configure Core0 with QSPI0
		`{0, 0},    /*SpiChannel_2*/`
		`{0, 0},    /*SpiChannel_1*/`
		`{0, 0},    /*SpiChannel_0*/`

1.2.7.2 Member: DataTransferLength

Table 124 DataTransferLength

Name	DataTransferLength	
Type	uint16	
Description	Indicates the number of data elements to be transmitted for a channel.	
Verification method	The member is generated as 0, if the channel SpiChannelType = EB The member is generated based on the configuration parameter SpiIbNBuffers when SpiChannelType = IB	
Example(s)	**Action**	**Generated output**

	Configure 3 IB channels
	SpiChannel_0/SpiDataWidth = 8
	SpiChannel_0/ SpiIbNBuffers = 10
	SpiChannel_1/SpiDataWidth = 16
	SpiChannel_1/ SpiIbNBuffers = 10
	SpiChannel_2/SpiDataWidth = 32
	SpiChannel_2/ SpiIbNBuffers
		`{0, 10},  /* for IB channels SpiChannel_2 */`
		`{40, 10},  /* for IB channels SpiChannel_1 */`
		`{60, 10},  /* for IB channels SpiChannel_0 */`

= 10	

Configure 3 channels, and mark all the channel types to EB
SpiChannelType= EB for all channels.
Configure Core0 with QSPI0
	`{0, 0},    /*SpiChannel_2*/`
	`{0, 0},    /*SpiChannel_1*/`
	`{0, 0},    /*SpiChannel_0*/`

1.2.8 Array: <SymbolicSequenceName>_ JobLinkPtr_Physical

Table 125 <SymbolicSequenceName>_ JobLinkPtr_Physical

Name	<SymbolicSequenceName>_ JobLinkPtr_Physical	
Type	Spi_JobType	
Description	Each array element represents the index for a job located under the Job configuration Spi_kJobConfig_Core<x> structure.	
Verification method	The array contains the index for each job linked for a sequence. This index is used to access the job configuration data of a particular job within the Core-job configuration Spi_kJobConfig_Core<x>.The last element of array is always SPI_JOB_DELIMITER which marks the end.	
Example(s)	**Action**	**Generated output**

	• Configure SpiSequence_0, SpiSequence_1
	• Assign SpiJob_0, SpiJob_1, SpiJob_2 to SpiSequence_0.
	• Assign SpiJob_3, SpiJob_4, SpiJob_5 to SpiSequence_1.
	• Configure QSPI0 and assign to Core0
	• Configure QSPI1 and assign to Core1
	• Configure SpiJob_0, SpiJob_1, SpiJob_2, SpiJob_3, SpiJob_4, SpiJob_5
	• Configure 12 channels
	• Assign SpiChannel_0, SpiChannel_1 for SpiJob_0
	• Assign SpiChannel_2, SpiChannel_3 for SpiJob_1
	• Assign SpiChannel_4, SpiChannel_5 for SpiJob_2
	• Assign SpiChannel_6, SpiChannel_7 for SpiJob_3
	• Assign SpiChannel_8, SpiChannel_9 for SpiJob_4
	• Assign SpiChannel_10, SpiChannel_11 for SpiJob_5
	• Drive the
		`static const Spi_JobType SpiSequence_0_JobLinkPtr_Physical[] =`
		`{`
		`  0U,        /* Physical index value for Job SpiJob_0 Job ID 0 */`
		
		`  1U,        /* Physical index value for Job SpiJob_1 Job ID 1 */`
		
		`  2U,        /* Physical index value for Job SpiJob_2 Job ID 2 */`
		
		`  SPI_JOB_DELIMITER`
		`};`
		
		`static const Spi_JobType SpiSequence_1_JobLinkPtr_Physical[] =`
		`{`
		`  0U,        /* Physical index value for Job SpiJob_3 Job ID 3 */`
		
		`  1U,        /* Physical index value for Job SpiJob_4 Job ID 4 */`

SpiExternalDevice_0 using SpiJob_0, SpiJob_1, SpiJob_2 and uses SpiHwUnit = QSPI0 • Drive the SpiExternalDevice_1 using SpiJob_3, SpiJob_4, SpiJob_5 and uses SpiHwUnit = QSPI1	2U, /* Physical index value for Job SpiJob_5 Job ID 5 */ SPI_JOB_DELIMITER };

• Configure SpiSequence_0, SpiSequence_1
• Assign SpiJob_0, SpiJob_1, SpiJob_2 to SpiSequence_0.
• Assign SpiJob_3, SpiJob_4, SpiJob_5 to SpiSequence_1.
• Configure QSPI0 and assign to Core0
• Configure QSPI1 and assign to Core0
• Configure SpiJob_0, SpiJob_1, SpiJob_2, SpiJob_3, SpiJob_4, SpiJob_5
• Configure 12 channels
• Assign SpiChannel_0, SpiChannel_1 for SpiJob_0
• Assign SpiChannel_2, SpiChannel_3 for SpiJob_1
• Assign SpiChannel_4, SpiChannel_5 for SpiJob_2
• Assign SpiChannel_6, SpiChannel_7 for SpiJob_3
• Assign SpiChannel_8, SpiChannel_9 for SpiJob_4
• Assign SpiChannel_10, SpiChannel_11 for SpiJob_5
• Drive the SpiExternalDevice_0 using SpiJob_0, SpiJob_1, SpiJob_2 and uses SpiHwUnit = QSPI0
• Drive the SpiExternalDevice_1 using SpiJob_3, SpiJob_4, SpiJob_5 and uses SpiHwUnit = QSPI1
	static const Spi_JobType SpiSequence_0_JobLinkPtr_Physical[] =
	{
	0U, /* Physical index value for Job SpiJob_0 Job ID 0 */
	1U, /* Physical index value for Job SpiJob_1 Job ID 1 */
	2U, /* Physical index value for Job SpiJob_2 Job ID 2 */
	SPI_JOB_DELIMITER
	};
	static const Spi_JobType SpiSequence_1_JobLinkPtr_Physical[] =
	{
	3U, /* Physical index value for Job SpiJob_3 Job ID 3 */
	4U, /* Physical index value for Job SpiJob_4 Job ID 4 */
	5U, /* Physical index value for Job SpiJob_5 Job ID 5 */
	SPI_JOB_DELIMITER
	};

1.2.9 Array: <SymbolicJobName>_ ChannelLinkPtr_Physical

Table 126 <SymbolicJobName>_ ChannelLinkPtr_Physical

Name	<SymbolicJobName>_ ChannelLinkPtr_Physical	
Type	Spi_ChannelType	
Description	Each array element represents the index for a channel located under the channel configuration Spi_kChannelConfig_Core<x> structure.	
Verification method	The array contains the index for each channel linked for a job. This index is used to access the channel configuration data of a particular channel within the Core-channel configuration Spi_kChannelConfig_Core<x>. The last element of array is always SPI_CHANNEL_DELIMITER which marks the end.	
Example(s)	**Action**	**Generated output**

	• Configure SpiSequence_0, SpiSequence_1
	• Assign SpiJob_0, SpiJob_1, SpiJob_2 to SpiSequence_0.
	• Assign SpiJob_3, SpiJob_4, SpiJob_5 to SpiSequence_1.
	• Configure QSPI0 and assign to Core0
	• Configure QSPI1 and assign to Core1
	• Configure SpiJob_0, SpiJob_1, SpiJob_2, SpiJob_3, SpiJob_4, SpiJob_5
	• Configure 12 channels
	• Assign SpiChannel_0, SpiChannel_1 for SpiJob_0
	• Assign SpiChannel_2, SpiChannel_3 for SpiJob_1
	• Assign SpiChannel_4, SpiChannel_5 for SpiJob_2
	• Assign SpiChannel_6, SpiChannel_7 for SpiJob_3
	• Assign SpiChannel_8, SpiChannel_9 for SpiJob_4
	• Assign SpiChannel_10, SpiChannel_11 for SpiJob_5
		/* Linked list for the channel[s] assigned to the job[s] Physical */
		
		static const Spi_ChannelType SpiJob_0_ChannelLinkPtr_Physical[] =
		{
		  5U, /* Physical index value for Channel SpiChannel_0 Channel ID 0 */
		
		  4U, /* Physical index value for Channel SpiChannel_1 Channel ID 1 */
		
		  SPI_CHANNEL_DELIMITER
		};
		
		static const Spi_ChannelType SpiJob_1_ChannelLinkPtr_Physical[] =
		{
		  3U, /* Physical index value for Channel SpiChannel_2 Channel ID 2 */
		
		  2U, /* Physical index value for Channel SpiChannel_3 Channel ID 3 */
		
		  SPI_CHANNEL_DELIMITER
		};

• Drive the SpiExternalDevice_0 using SpiJob_0, SpiJob_1, SpiJob_2 and uses SpiHwUnit = QSPI0
• Drive the SpiExternalDevice_1 using SpiJob_3, SpiJob_4, SpiJob_5 and uses SpiHwUnit = QSPI1
	`static const Spi_ChannelType SpiJob_2_ChannelLinkPtr_Physical[] =`
	`{`
	`  1U,         /* Physical index value for Channel SpiChannel_4 Channel ID 4 */`
	
	`  0U,         /* Physical index value for Channel SpiChannel_5 Channel ID 5 */`
	
	`  SPI_CHANNEL_DELIMITER`
	`};`
	
	`static const Spi_ChannelType SpiJob_3_ChannelLinkPtr_Physical[] =`
	`{`
	`  5U,         /* Physical index value for Channel SpiChannel_6 Channel ID 6 */`
	
	`  4U,         /* Physical index value for Channel SpiChannel_7 Channel ID 7 */`
	
	`  SPI_CHANNEL_DELIMITER`
	`};`
	
	`static const Spi_ChannelType SpiJob_4_ChannelLinkPtr_Physical[] =`
	`{`
	`  3U,         /* Physical index value for Channel SpiChannel_8 Channel ID 8 */`
	
	`  2U,         /* Physical index value for Channel SpiChannel_9 Channel ID 9 */`
	
	`  SPI_CHANNEL_DELIMITER`
	`};`
	
	`static const Spi_ChannelType`

	SpiJob_5_ChannelLinkPtr_Physical[] =
	{
	  1U, /* Physical index value for Channel SpiChannel_10 Channel ID 10 */
	
	  0U, /* Physical index value for Channel SpiChannel_11 Channel ID 11 */
	
	  SPI_CHANNEL_DELIMITER
	};

• Configure SpiSequence_0, SpiSequence_1
• Assign SpiJob_0, SpiJob_1, SpiJob_2 to SpiSequence_0.
• Assign SpiJob_3, SpiJob_4, SpiJob_5 to SpiSequence_1.
• Configure QSPI0 and assign to Core0
• Configure QSPI1 and assign to Core0
• Configure SpiJob_0, SpiJob_1, SpiJob_2, SpiJob_3, SpiJob_4, SpiJob_5
• Configure 12 channels
• Assign SpiChannel_0, SpiChannel_1 for SpiJob_0
• Assign SpiChannel_2, SpiChannel_3 for SpiJob_1
• Assign SpiChannel_4, SpiChannel_5 for SpiJob_2
• Assign SpiChannel_6, SpiChannel_7 for SpiJob_3
• Assign SpiChannel_8, SpiChannel_9 for SpiJob_4
• Assign SpiChannel_10, SpiChannel_11 for
	static const Spi_ChannelType SpiJob_0_ChannelLinkPtr_Physical[] =
	{
	  5U, /* Physical index value for Channel SpiChannel_0 Channel ID 0 */
	
	  4U, /* Physical index value for Channel SpiChannel_1 Channel ID 1 */
	
	  SPI_CHANNEL_DELIMITER
	};
	
	static const Spi_ChannelType SpiJob_1_ChannelLinkPtr_Physical[] =
	{
	  3U, /* Physical index value for Channel SpiChannel_2 Channel ID 2 */
	
	  2U, /* Physical index value for Channel SpiChannel_3 Channel ID 3 */
	
	  SPI_CHANNEL_DELIMITER
	};
	
	static const Spi_ChannelType SpiJob_2_ChannelLinkPtr_Physical[] =
	{

	SpiJob_5
	• Drive the SpiExternalDevice_0 using SpiJob_0, SpiJob_1, SpiJob_2 and uses SpiHwUnit = QSPI0
	• Drive the SpiExternalDevice_1 using SpiJob_3, SpiJob_4, SpiJob_5 and uses SpiHwUnit = QSPI1
		`  1U,         /* Physical index value for Channel SpiChannel_4 Channel ID 4 */`
		
		`  0U,         /* Physical index value for Channel SpiChannel_5 Channel ID 5 */`
		
		`  SPI_CHANNEL_DELIMITER`
		`};`
		
		`static const Spi_ChannelType SpiJob_3_ChannelLinkPtr_Physical[] =`
		`{`
		`  11U,         /* Physical index value for Channel SpiChannel_6 Channel ID 6 */`
		
		`  10U,         /* Physical index value for Channel SpiChannel_7 Channel ID 7 */`
		
		`  SPI_CHANNEL_DELIMITER`
		`};`
		
		`static const Spi_ChannelType SpiJob_4_ChannelLinkPtr_Physical[] =`
		`{`
		`  9U,         /* Physical index value for Channel SpiChannel_8 Channel ID 8 */`
		
		`  8U,         /* Physical index value for Channel SpiChannel_9 Channel ID 9 */`
		
		`  SPI_CHANNEL_DELIMITER`
		`};`
		
		`static const Spi_ChannelType SpiJob_5_ChannelLinkPtr_Physical[] =`
		`{`
		`  7U,         /* Physical index value for Channel SpiChannel_10 Channel ID 10`

		`*/`
		`  6U,        /* Physical index value for Channel SpiChannel_11 Channel ID 11 */`
		`  SPI_CHANNEL_DELIMITER`
		`};`

1.2.10 Array: SequenceLookupIndex[_<variant>]

Table 127 SequenceLookupIndex[_<variant>]

Name	SequenceLookupIndex[_<variant>]	
Type	uint8	
Description	Each array element represents the index for a sequence located under the sequence configuration Spi_kSequenceConfig_Core<x> structure.	
Verification method	The array contains the index for each sequence configured. This index is used to access the sequence configuration data of a particular sequence within the Core-Sequence configuration Spi_kSequenceConfig_Core<x>.	
Example(s)	**Action**	**Generated output**

	• Configure SpiSequence_0, SpiSequence_1
	• Assign SpiJob_0, SpiJob_1, SpiJob_2 to SpiSequence_0.
	• Assign SpiJob_3, SpiJob_4, SpiJob_5 to SpiSequence_1.
	• Configure QSPI0 and assign to Core0
	• Configure QSPI1 and assign to Core1
	• Configure SpiJob_0, SpiJob_1, SpiJob_2, SpiJob_3, SpiJob_4, SpiJob_5
	• Configure 12 channels
	• Assign SpiChannel_0, SpiChannel_1 for SpiJob_0
	• Assign SpiChannel_2, SpiChannel_3 for SpiJob_1
	• Assign SpiChannel_4, SpiChannel_5 for SpiJob_2
		`static const uint8 SequenceLookupIndex[2] =`
		`{`
		`    /* Physical index value for Sequence SpiSequence_0 Sequence ID 0 */`
		`    0U,`
		`    /* Physical index value for Sequence SpiSequence_1 Sequence ID 1 */`
		`    0U`
		`};`

• Assign SpiChannel_6, SpiChannel_7 for SpiJob_3 • Assign SpiChannel_8, SpiChannel_9 for SpiJob_4 • Assign SpiChannel_10, SpiChannel_11 for SpiJob_5 • Drive the SpiExternalDevice_0 using SpiJob_0, SpiJob_1, SpiJob_2 and uses SpiHwUnit = QSPI0 • Drive the SpiExternalDevice_1 using SpiJob_3, SpiJob_4, SpiJob_5 and uses SpiHwUnit = QSPI1	

• Configure SpiSequence_0, SpiSequence_1
• Assign SpiJob_0, SpiJob_1, SpiJob_2 to SpiSequence_0.
• Assign SpiJob_3, SpiJob_4, SpiJob_5 to SpiSequence_1.
• Configure QSPI0 and assign to Core0
• Configure QSPI1 and assign to Core0
• Configure SpiJob_0, SpiJob_1, SpiJob_2, SpiJob_3, SpiJob_4, SpiJob_5
• Configure 12 channels
• Assign SpiChannel_0, SpiChannel_1 for SpiJob_0
• Assign SpiChannel_2, SpiChannel_3 for SpiJob_1
• Assign SpiChannel_4, SpiChannel_5 for SpiJob_2
• Assign SpiChannel_6,
	`static const uint8 SequenceLookupIndex[2] =`
	`{`
	`    /* Physical index value for Sequence SpiSequence_0 Sequence ID 0 */`
	`    0U,`
	`    /* Physical index value for Sequence SpiSequence_1 Sequence ID 1 */`
	`    1U`
	`};`

	SpiChannel_7 for SpiJob_3 • Assign SpiChannel_8, SpiChannel_9 for SpiJob_4 • Assign SpiChannel_10, SpiChannel_11 for SpiJob_5 • Drive the SpiExternalDevice_0 using SpiJob_0, SpiJob_1, SpiJob_2 and uses SpiHwUnit = QSPI0 • Drive the SpiExternalDevice_1 using SpiJob_3, SpiJob_4, SpiJob_5 and uses SpiHwUnit = QSPI1	

1.2.11 Array: JobLookupIndex[_<variant>]

Table 128 JobLookupIndex[_<variant>]

Name	JobLookupIndex[_<variant>]	
Type	uint16	
Description	Each array element represents the index for a job located under the job configuration Spi_kJobConfig_Core<x> structure.	
Verification method	The array contains the index for each Job configured. This index is used to access the job configuration data of a particular job within the Core-job configuration Spi_kJobConfig_Core<x>.	
Example(s)	**Action**	**Generated output**

	• Configure SpiSequence_0, SpiSequence_1
	• Assign SpiJob_0, SpiJob_1, SpiJob_2 to SpiSequence_0.
	• Assign SpiJob_3, SpiJob_4, SpiJob_5 to SpiSequence_1.
	• Configure QSPI0 and assign to Core0
	• Configure QSPI1 and assign to Core1
	• Configure SpiJob_0, SpiJob_1, SpiJob_2, SpiJob_3, SpiJob_4,
		`static const uint16 JobLookupIndex[6] =`
		`{`
		
		`    /* Physical index value for Job SpiJob_0 Job ID 0 */`
		`    0U,`
		
		`    /* Physical index value for Job SpiJob_1 Job ID 1 */`
		`    1U,`
		
		`    /* Physical index value for Job SpiJob_2 Job ID 2 */`

SpiJob_5
- Configure 12 channels
- Assign SpiChannel_0, SpiChannel_1 for SpiJob_0
- Assign SpiChannel_2, SpiChannel_3 for SpiJob_1
- Assign SpiChannel_4, SpiChannel_5 for SpiJob_2
- Assign SpiChannel_6, SpiChannel_7 for SpiJob_3
- Assign SpiChannel_8, SpiChannel_9 for SpiJob_4
- Assign SpiChannel_10, SpiChannel_11 for SpiJob_5
- Drive the SpiExternalDevice_0 using SpiJob_0, SpiJob_1, SpiJob_2 and uses SpiHwUnit = QSPI0
- Drive the SpiExternalDevice_1 using SpiJob_3, SpiJob_4, SpiJob_5 and uses SpiHwUnit = QSPI1
	`    2U,`
	
	`    /* Physical index value for Job SpiJob_3 Job ID 3 */`
	`    0U,`
	
	`    /* Physical index value for Job SpiJob_4 Job ID 4 */`
	`    1U,`
	
	`    /* Physical index value for Job SpiJob_5 Job ID 5 */`
	`    2U`
	`};`

- Configure SpiSequence_0, SpiSequence_1
- Assign SpiJob_0, SpiJob_1, SpiJob_2 to SpiSequence_0.
- Assign SpiJob_3, SpiJob_4, SpiJob_5 to SpiSequence_1.
- Configure QSPI0 and assign to Core0
- Configure QSPI1 and assign to Core0
- Configure SpiJob_0, SpiJob_1, SpiJob_2, SpiJob_3, SpiJob_4, SpiJob_5
	`static const uint16 JobLookupIndex[6] =`
	`{`
	
	`    /* Physical index value for Job SpiJob_0 Job ID 0 */`
	`    0U,`
	
	`    /* Physical index value for Job SpiJob_1 Job ID 1 */`
	`    1U,`
	
	`    /* Physical index value for Job SpiJob_2 Job ID 2 */`
	`    2U,`

	• Configure 12 channels
	• Assign SpiChannel_0, SpiChannel_1 for SpiJob_0
	• Assign SpiChannel_2, SpiChannel_3 for SpiJob_1
	• Assign SpiChannel_4, SpiChannel_5 for SpiJob_2
	• Assign SpiChannel_6, SpiChannel_7 for SpiJob_3
	• Assign SpiChannel_8, SpiChannel_9 for SpiJob_4
	• Assign SpiChannel_10, SpiChannel_11 for SpiJob_5
	• Drive the SpiExternalDevice_0 using SpiJob_0, SpiJob_1, SpiJob_2 and uses SpiHwUnit = QSPI0
	• Drive the SpiExternalDevice_1 using SpiJob_3, SpiJob_4, SpiJob_5 and uses SpiHwUnit = QSPI1
		`/* Physical index value for Job SpiJob_3 Job ID 3 */`
		`3U,`
		
		`/* Physical index value for Job SpiJob_4 Job ID 4 */`
		`4U,`
		
		`/* Physical index value for Job SpiJob_5 Job ID 5 */`
		`5U`
		`};`

1.2.12 Array: ChannelLookupIndex[_<variant>]

Table 129 ChannelLookupIndex[_<variant>]

Name	ChannelLookupIndex[_<variant>]	
Type	uint8	
Description	Each array element represents the index for a channel located under the channel configuration Spi_kChannelConfig_Core<x> structure.	
Verification method	The array contains the index for each Channel configured. This index is used to access the Channel configuration data of a particular channel within the Core-channel configuration Spi_kChannelConfig_Core<x>.	
Example(s)	**Action**	**Generated output**

	• Configure SpiSequence_0, SpiSequence_1
	• Assign SpiJob_0,
		`static const uint8 ChannelLookupIndex[12] =`
		`{`

SpiJob_1, SpiJob_2 to SpiSequence_0. • Assign SpiJob_3, SpiJob_4, SpiJob_5 to SpiSequence_1. • Configure QSPI0 and assign to Core0 • Configure QSPI1 and assign to Core1 • Configure SpiJob_0, SpiJob_1, SpiJob_2, SpiJob_3, SpiJob_4, SpiJob_5 • Configure 12 channels • Assign SpiChannel_0, SpiChannel_1 for SpiJob_0 • Assign SpiChannel_2, SpiChannel_3 for SpiJob_1 • Assign SpiChannel_4, SpiChannel_5 for SpiJob_2 • Assign SpiChannel_6, SpiChannel_7 for SpiJob_3 • Assign SpiChannel_8, SpiChannel_9 for SpiJob_4 • Assign SpiChannel_10, SpiChannel_11 for SpiJob_5 • Drive the SpiExternalDevice_0 using SpiJob_0, SpiJob_1, SpiJob_2 and uses SpiHwUnit = QSPI0 • Drive the SpiExternalDevice_1 using SpiJob_3, SpiJob_4, SpiJob_5 and uses SpiHwUnit = QSPI1	/* Physical index value for channel SpiChannel_0 channel ID 0 */ 5U, /* Physical index value for channel SpiChannel_1 channel ID 1 */ 4U, /* Physical index value for channel SpiChannel_2 channel ID 2 */ 3U, /* Physical index value for channel SpiChannel_3 channel ID 3 */ 2U, /* Physical index value for channel SpiChannel_4 channel ID 4 */ 1U, /* Physical index value for channel SpiChannel_5 channel ID 5 */ 0U, /* Physical index value for channel SpiChannel_6 channel ID 6 */ 5U, /* Physical index value for channel SpiChannel_7 channel ID 7 */ 4U, /* Physical index value for channel SpiChannel_8 channel ID 8 */ 3U, /* Physical index value for channel SpiChannel_9 channel ID 9 */ 2U, /* Physical index value for channel

	SpiChannel_10 channel ID 10 */
	    1U,
	
	    /* Physical index value for channel SpiChannel_11 channel ID 11 */
	    0U
	};

• Configure SpiSequence_0, SpiSequence_1 • Assign SpiJob_0, SpiJob_1, SpiJob_2 to SpiSequence_0. • Assign SpiJob_3, SpiJob_4, SpiJob_5 to SpiSequence_1. • Configure QSPI0 and assign to Core0 • Configure QSPI1 and assign to Core0 • Configure SpiJob_0, SpiJob_1, SpiJob_2, SpiJob_3, SpiJob_4, SpiJob_5 • Configure 12 channels • Assign SpiChannel_0, SpiChannel_1 for SpiJob_0 • Assign SpiChannel_2, SpiChannel_3 for SpiJob_1 • Assign SpiChannel_4, SpiChannel_5 for SpiJob_2 • Assign SpiChannel_6, SpiChannel_7 for SpiJob_3 • Assign SpiChannel_8, SpiChannel_9 for SpiJob_4 • Assign SpiChannel_10, SpiChannel_11 for SpiJob_5 • Drive the SpiExternalDevice_0 using SpiJob_0, SpiJob_1, SpiJob_2 and uses SpiHwUnit =	static const uint8 ChannelLookupIndex[12] = { /* Physical index value for channel SpiChannel_0 channel ID 0 */ 5U, /* Physical index value for channel SpiChannel_1 channel ID 1 */ 4U, /* Physical index value for channel SpiChannel_2 channel ID 2 */ 3U, /* Physical index value for channel SpiChannel_3 channel ID 3 */ 2U, /* Physical index value for channel SpiChannel_4 channel ID 4 */ 1U, /* Physical index value for channel SpiChannel_5 channel ID 5 */ 0U, /* Physical index value for channel SpiChannel_6 channel ID 6 */ 11U, /* Physical index value for channel SpiChannel_7 channel ID 7 */

	QSPI0
	• Drive the SpiExternalDevice_1 using SpiJob_3, SpiJob_4, SpiJob_5 and uses SpiHwUnit = QSPI1
		`10U,`
		
		`/* Physical index value for channel SpiChannel_8 channel ID 8 */`
		`9U,`
		
		`/* Physical index value for channel SpiChannel_9 channel ID 9 */`
		`8U,`
		
		`/* Physical index value for channel SpiChannel_10 channel ID 10 */`
		`7U,`
		
		`/* Physical index value for channel SpiChannel_11 channel ID 11 */`
		`6U`
		`};`

1.2.13 Array: <SymbolicSequenceName>_SeqSharePtr

Table 130 <SymbolicSequenceName>_SeqSharePtr

Name	<SymbolicSequenceName>_SeqSharePtr	
Type	Spi_SequenceType	
Description	Each array element represents the sequence ID with which the given sequence is sharing at least one job.	
Verification method	The array is generated only when SpiLevelDelivered = 1 or 2. The array is generated with valid sequence ID which is sharing the job with given sequence. The last element of array is always SPI_SEQUENCE_DELIMITER which marks the end. If SpiLevelDelivered = 0, the array is not generated. If there are no jobs shared, the array is generated with only the last element.	
Example(s)	**Action**	**Generated output**

	• SpiLevelDelivered = 1 or 2
	• Configure SpiSequence_0, SpiSequence_1
	• Assign SpiJob_0, SpiJob_1, SpiJob_2 to SpiSequence_0.
	• Assign SpiJob_2, SpiJob_3, SpiJob_4, SpiJob_5 to SpiSequence_1.
		`/* Linked list of sequence[s] with Job[s] shared */`
		`static const Spi_SequenceType SpiSequence_0_SeqSharePtr[] =`
		`{`
		
		`SpiConf_SpiSequence_SpiSequence_1,`
		`SPI_SEQUENCE_DELIMITER`
		`};`
		`static const Spi_SequenceType`

• Configure QSPI0 and assign to Core0
• Configure 12 channels
• Assign SpiChannel_0, SpiChannel_1 for SpiJob_0
• Assign SpiChannel_2, SpiChannel_3 for SpiJob_1
• Assign SpiChannel_4, SpiChannel_5 for SpiJob_2
• Assign SpiChannel_6, SpiChannel_7 for SpiJob_3
• Assign SpiChannel_8, SpiChannel_9 for SpiJob_4
• Assign SpiChannel_10, SpiChannel_11 for SpiJob_5
• Drive the SpiExternalDevice_0 using SpiJob_0, SpiJob_1, SpiJob_2, SpiJob_3, SpiJob_4, SpiJob_5 and uses SpiHwUnit = QSPI0
• SpiJob_2 is shared between the sequences.
	`SpiSequence_1_SeqSharePtr[] =`
	`{`
	`  SpiConf_SpiSequence_SpiSequence_0,`
	`  SPI_SEQUENCE_DELIMITER`
	`};`

• Configure SpiSequence_0, SpiSequence_1
• Assign SpiJob_0, SpiJob_1, SpiJob_2 to SpiSequence_0.
• Assign SpiJob_3, SpiJob_4, SpiJob_5 to SpiSequence_1.
• Configure QSPI0 and assign to Core0
• Configure 12 channels
• Assign SpiChannel_0, SpiChannel_1 for SpiJob_0
• Assign SpiChannel_2, SpiChannel_3 for SpiJob_1
	`/* Linked list of sequence[s] with Job[s] shared  */`
	`static const Spi_SequenceType SpiSequence_0_SeqSharePtr[] =`
	`{`
	`  SPI_SEQUENCE_DELIMITER`
	`};`
	`static const Spi_SequenceType SpiSequence_1_SeqSharePtr[] =`
	`{`
	`  SPI_SEQUENCE_DELIMITER`
	`};`

• Assign SpiChannel_4, SpiChannel_5 for SpiJob_2 • Assign SpiChannel_6, SpiChannel_7 for SpiJob_3 • Assign SpiChannel_8, SpiChannel_9 for SpiJob_4 • Assign SpiChannel_10, SpiChannel_11 for SpiJob_5 • Drive the SpiExternalDevice_0 using SpiJob_0, SpiJob_1, SpiJob_2, SpiJob_3, SpiJob_4, SpiJob_5 and uses SpiHwUnit = QSPI0 • No Job sharing between the sequences.	
• SpiLevelDelivered = 0 • Configure SpiSequence_0, SpiSequence_1 • Assign SpiJob_0, SpiJob_1, SpiJob_2 to SpiSequence_0. • Assign SpiJob_2, SpiJob_3, SpiJob_4, SpiJob_5 to SpiSequence_1. • Configure QSPI0 and assign to Core0 • Configure 12 channels • Assign SpiChannel_0, SpiChannel_1 for SpiJob_0 • Assign SpiChannel_2, SpiChannel_3 for SpiJob_1 • Assign SpiChannel_4, SpiChannel_5 for SpiJob_2 • Assign SpiChannel_6, SpiChannel_7 for SpiJob_3 • Assign SpiChannel_8,	`Array is not generated.`

	SpiChannel_9 for SpiJob_4 • Assign SpiChannel_10, SpiChannel_11 for SpiJob_5 • Drive the SpiExternalDevice_0 using SpiJob_0, SpiJob_1, SpiJob_2, SpiJob_3, SpiJob_4, SpiJob_5 and uses SpiHwUnit = QSPI0 • SpiJob_2 is shared between the sequences.	

1.2.14 Function Pointer: SpiSeqEndNotification

Table 131 SpiSeqEndNotification

Name	SpiSeqEndNotification	
Type	void(*Spi_SeqEndNotification)(void)	
Description	User notification function to be called after sequence transmission.	
Verification method	If SpiLevelDelivered = 1 or 2 and SpiSequence/SpiSeqEndNotification contains function name, The function configured in 'SpiSequence/SpiSeqEndNotification' would be populated as a prototype with extern qualifier. Otherwise the prototype is not generated.	
Example(s)	**Action**	**Generated output**
	SpiSequence/ SpiSeqEndNotification = EEP_TEST_Notification	`extern void EEP_TEST_Notification(void);`
	SpiSequence/ SpiSeqEndNotification = 0xABCD	`The prototype is not generated.`
	If SpiSequence/ SpiSeqEndNotification not configured	`The prototype is not generated.`

1.2.15 Function Pointer: SpiJobEndNotification

Table 132 SpiJobEndNotification

Name	SpiJobEndNotification	
Type	void(*Spi_JobEndNotification)(void)	
Description	User notification function to be called after job transmission.	
Verification method	If SpiLevelDelivered = 1 or 2 and SpiJob/SpiJobEndNotification contains function name, The function configured in 'SpiJob/SpiJobEndNotification' would be populated as a prototype with extern qualifier. Otherwise the prototype is not generated.	
Example(s)	**Action**	**Generated output**

	SpiJob/ SpiJobEndNotification = EEP_TEST_Notification	extern void EEP_TEST_Notification(void);
	SpiJob/ SpiJobEndNotification = 0xABCD	The prototype is not generated.
	If SpiJob / SpiJobEndNotification not configured	The prototype is not generated.

Revision history

Major changes since the last revision

Date	Version	Description
2023-05-26	7.0	Released Version.
2023-05-26	6.1	-For the following Parameters Verification Method and Example(s) is updated. - 1.1.18.SPI_MULTICORE_ERROR_DETECT - 1.1.45 SPI_JOB_QUEUE_LENGTH_QSPIx - 1.2.3.7 HwModuleUsed - 1.2.3.8 u8Comm - 1.2.6.1 Member: ActiveChipSelectLevel - For the following Parameters Verification Method - 1.1.50 SPI_DMA_MAX_TCS_NUM_QSPI<x> - 1.2.4.3 BaudRateAndClockParam - 1.2.6.7 MasterReceivePortPin -In Section 1.2.4.4 IdleLeadTrailDelay updated Generated Output in Example(s). -Removed the Section 1.3 File: Spi[_<variant>] PBcfg.h -Changed DEM to Production Error where applicable in sections: 1.1.8, 1.1.9, 1.1.10.
2022-07-08	6.0	Released Version
2022-06-30	5.1	In section 1.2.4.4 IdleLeadTrailDelay Verification method and Example(s) is updated
2021-03-24	5.0	Front page aligned as per template. Released version.
2021-03-23	4.1	Macro SPI_MAX_HW_UNIT information is updated.
2020-08-06	4.0	Released Version. Review comments fixed: Spi_kQspiHwConfigQSPI<x> updated.
2020-08-06	3.1	- Spi driver chapter moved from MC-ISAR_TC3xx_Config_Verification_Manual_BASIC.pdf to this document. - Added the following new macros: SPI_RUNTIME_ERROR_DETECT SPI_CONTROL_LOOPBACK_API SPI_IB_BUFFER_SIZE_CORE<x> Rmoved the following unused macros: SPI_JOB_STATUS_ARRAY_INDEX SPI_SEQUENCE_STATUS_ARRAY_INDEX SPI_MAX_SEQUENCE_QSPI SPI_SYNC_IB_BUFFER_SIZE_QSPI<x> SPI_ASYNC_IB_BUFFER_SIZE_QSPI<x> SPI_SYNC_IB_BUFFER_SIZE_CORE<x> SPI_ASYNC_IB_BUFFER_SIZE_CORE<x> SPI_NUM_ASYNC_IB_CHANNELS_QSPI<x>

Revision history

Date	Version	Description
		SPI_NUM_SYNC_IB_CHANNELS_QSPI<x> SPI_NUM_IB_CHANNELS_QSPI<x> SPI_WRITE_LOCK_INDEX SPI_WRITE_LOCK_INDEX_QSPI<x> SPI_NUM_EB_CHANNELS_QSPI<x> SPI_NUM_ASYNC_QSPI<x>_MASTER SPI_NUM_SYNC_QSPI<x>_MASTER SPI_NUM_QSPI<x>_MASTER
2019-07-16	3.0	Fixed review comments. Released.
2019-07-09	2.1	Added strcture members ExternalDemuxEnabled, StrobeDelay to Spi_QspiHwConfigType
2019-02-27	1.10.0_2.0	Added PBcfg.h
2019-02-26	1.10.0_1.0	Released Version.
2019-02-22	1.10.0_0.1	Initial Version

Trademarks

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

Edition 2023-05-26
Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
Doc_Number

IMPORTANT NOTICE

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

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

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

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

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS

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

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