
User Manual

for MPC563XM SPI Driver

Document Number: UM14SPIASR3.0R2.0.0

Rev. 1.2





Contents

Section Number	Title	Page
Chapter 1		
Revision History		
Chapter 2		
Introduction		
2.1	Supported Derivatives.....	11
2.2	Overview.....	11
2.3	About this Manual.....	11
2.4	Acronyms and Definitions.....	12
2.5	Reference List.....	12
Chapter 3		
Driver		
3.1	Requirements.....	15
3.2	Driver Design Summary.....	15
3.3	Deviation from Requirements.....	17
3.4	Function Definitions.....	18
3.4.1	Function Spi_AsyncTransmit.....	18
3.4.2	Function Spi_Cancel.....	19
3.4.3	Function Spi_DeInit.....	19
3.4.4	Function Spi_GetAsyncStatus.....	20
3.4.5	Function Spi_GetHWUnitStatus.....	20
3.4.6	Function Spi_GetJobResult.....	21
3.4.7	Function Spi_GetSequenceResult.....	21
3.4.8	Function Spi_GetStatus.....	22
3.4.9	Function Spi_GetVersionInfo.....	22
3.4.10	Function Spi_Init.....	23
3.4.11	Function Spi_MainFunction_Driving.....	23
3.4.12	Function Spi_ReadIB.....	24
3.4.13	Function Spi_SetAsyncMode.....	24

Section Number	Title	Page
3.4.14	Function Spi_SetupEB.....	25
3.4.15	Function Spi_SyncTransmit.....	26
3.4.16	Function Spi_WriteIB.....	26
3.4.17	Function Spi_SetHWUnitAsyncMode.....	27
3.4.18	Function Spi_SetClockMode.....	28
3.4.19	Function Spi_TSB_ASDR_DataUpdate.....	28
3.4.20	Function Spi_TSBBstart.....	29
3.4.21	Function Spi_TSBBstop.....	30
3.4.22	Function Index.....	30
3.5	Enum Definitions.....	31
3.5.1	Enumeration Spi_DualClockMode.....	31
3.5.2	Enumeration Spi_JobResultType.....	31
3.5.3	Enumeration Spi_SeqResultType.....	32
3.5.4	Enumeration Spi_StatusType.....	32
3.5.5	Enumeration Spi_AsyncModeType.....	32
3.6	Structure Definitions.....	33
3.6.1	Structure Spi_HWUnitConfig.....	33
3.6.2	Structure Spi_ChannelConfig.....	33
3.6.3	Structure Spi_JobConfig.....	34
3.6.4	Structure Spi_SequenceConfig.....	35
3.6.5	Structure Spi_TSBBConfig.....	35
3.7	Define Definitions.....	36
3.7.1	Define SPI_ASYNCTRANSMIT_ID.....	36
3.7.2	Define SPI_CANCEL_ID.....	36
3.7.3	Define SPI_DEINIT_ID.....	36
3.7.4	Define SPI_GETHWUNITSTATUS_ID.....	36
3.7.5	Define SPI_GETJOBRESULT_ID.....	36
3.7.6	Define SPI_GETSEQUENCERESULT_ID.....	37
3.7.7	Define SPI_GETSTATUS_ID.....	37

Section Number	Title	Page
3.7.8	Define SPI_GETVERSIONINFO_ID.....	37
3.7.9	Define SPI_INIT_ID.....	37
3.7.10	Define SPI_MAINFUNCTION_DRIVING_ID.....	37
3.7.11	Define SPI_MODULE_ID.....	37
3.7.12	Define SPI_READIB_ID.....	38
3.7.13	Define SPI_SETASYNCMODE_ID.....	38
3.7.14	Define SPI_SETCLOCKMODE_ID.....	38
3.7.15	Define SPI_SETHWUNITASYNCMODE_ID.....	38
3.7.16	Define SPI_SETUPEB_ID.....	38
3.7.17	Define SPI_SYNCTRANSMIT_ID.....	38
3.7.18	Define SPI_VENDOR_ID.....	39
3.7.19	Define SPI_WRITEIB_ID.....	39
3.7.20	Define SPI_E_ALREADY_INITIALIZED.....	39
3.7.21	Define SPI_E_CONFIG_OUT_OF_RANGE.....	39
3.7.22	Define SPI_E_JOB_EMPTY.....	39
3.7.23	Define SPI_E_PARAM_CHANNEL.....	39
3.7.24	Define SPI_E_PARAM_EB_UNIT.....	40
3.7.25	Define SPI_E_PARAM_JOB.....	40
3.7.26	Define SPI_E_PARAM_LENGTH.....	40
3.7.27	Define SPI_E_PARAM_SEQ.....	40
3.7.28	Define SPI_E_PARAM_UNIT.....	40
3.7.29	Define SPI_E_SEQ_EMPTY.....	40
3.7.30	Define SPI_E_SEQ_IN_PROCESS.....	41
3.7.31	Define SPI_E_SEQ_PENDING.....	41
3.7.32	Define SPI_E_UNINIT.....	41
3.7.33	Define SPI_SETTSBMODE_ID.....	41
3.8	Symbolic Names DISCLAIMER.....	41

Section Number	Title	Page
3.9	Configuration Parameters.....	41
3.9.1	Pre-Compile Parameters.....	42
3.9.1.1	DualReceiverSupport.....	44
3.9.1.2	SpiCancelApi.....	44
3.9.1.3	SpiChannelBuffersAllowed.....	44
3.9.1.4	SpiDevErrorDetect	45
3.9.1.5	SpiHWStatusAPI.....	45
3.9.1.6	SpiInterruptibleSeqAllowed.....	45
3.9.1.7	SpiLevelDelivered.....	46
3.9.1.8	SpiLevelDelivered.....	46
3.9.1.9	SpiClockReference.....	47
3.9.1.10	SpiGlobalDmaEnable.....	47
3.9.1.11	SpiSyncTransmitTimeout.....	48
3.9.1.12	SpiOptimizeOneJobSequences.....	48
3.9.1.13	SpiOptimizedChannelsNumber.....	49
3.9.1.14	SpiOptimizedSeqNumber.....	49
3.9.1.15	SpiEnableMultiSyncTransmit.....	50
3.9.1.16	SpiEnableHWUnitAsyncMode	50
3.9.1.17	SpiEnableDualClockMode.....	50
3.9.1.18	SpiAlternateClockRef.....	51
3.9.1.19	SpiAllowBigSizeCollections.....	52
3.9.1.20	SpiJobStartNotificationEnable.....	52
3.9.1.21	SpiTSBModeSupport.....	52
3.9.1.22	TSBModeEnable.....	52
3.9.2	Post-Build parameters.....	53
3.9.2.1	SpiPhyUnitMapping.....	56
3.9.2.2	SpiPhyUnitSync.....	57
3.9.2.3	SpiPhyUnitAsyncMethod.....	57
3.9.2.4	SpiPhyTxDmaChannel.....	58

Section Number	Title	Page
3.9.2.5	SpiPhyTxDmaChannelAux.....	59
3.9.2.6	SpiPhyRxDmaChannel.....	59
3.9.2.7	ChannelAssignment.....	60
3.9.2.8	SpiMaxChannel.....	61
3.9.2.9	SpiChannelId.....	62
3.9.2.10	SpiChannelType.....	62
3.9.2.11	SpiDataWidth.....	62
3.9.2.12	SpiDefaultData.....	63
3.9.2.13	SpiEbMaxlength.....	63
3.9.2.14	SpiIbNBuffers.....	64
3.9.2.15	SpiTransferStart.....	65
3.9.2.16	SpiBaudRate.....	65
3.9.2.17	SpiCs.....	66
3.9.2.18	SpiCsPolarity.....	66
3.9.2.19	SpiDataShiftEdge.....	67
3.9.2.20	SpiEnableCs.....	68
3.9.2.21	SpiShiftClockIdleLevel.....	69
3.9.2.22	SpiTimeClk2Cs.....	69
3.9.2.23	SpiTimeCs2Clk.....	70
3.9.2.24	SpiTimeCs2Cs.....	70
3.9.2.25	SpiCsContinuous.....	71
3.9.2.26	SpiMaxJob.....	72
3.9.2.27	SpiHwUnit.....	73
3.9.2.28	SpiJobId.....	73
3.9.2.29	SpiJobPriority.....	73
3.9.2.30	DeviceAssignment.....	74
3.9.2.31	SpiMaxSequence.....	75
3.9.2.32	SpiInterruptibleSequence.....	76
3.9.2.33	SpiSeqEndNotification.....	76

Section Number	Title	Page
3.9.2.34	SpiSequenceId.....	76
3.9.2.35	JobAssignment.....	77
3.9.2.36	SpiJobEndNotification.....	77
3.9.2.37	SpiJobstartNotification.....	78
3.9.2.38	DsiCsIdentifier.....	79
3.9.2.39	SecondaryDsiCsIdentifier.....	80
3.9.2.40	SecondaryFrameSize.....	81
3.9.2.41	TransmitDataSource.....	82
3.9.2.42	TSBFrameSize.....	83
3.9.2.43	ChangeInDataTransfer.....	84

Chapter 4

How to use the same channel in multiple post build Configurations

Chapter 5

DEM events reported by SPI driver

5.1	DEM events	89
-----	------------------	----

Chapter 1

Revision History

Table 1-1. Document Change History

Date	Version	Changed by	Change description
10-Feb-2011	1.0	Srikanth M.S	Initial version
13-Apr-2011	1.1	Rutuja Bichile	Updated for Timed Serial Bus Support
20-Dec-2011	1.2	Subramanya M Naik	Updated for MPC5634M RTM 2.0.0 Release



Chapter 2

Introduction

This User's Manual describes Freescale Semiconductor AUTOSAR Serial Peripheral interface (SPI) driver.

AUTOSAR SPI driver configuration parameters and deviations from the specification are described in SPI Driver chapter. AUTOSAR SPI driver requirements and APIs are described in the AUTOSAR SPI driver software specification document (See Table 2.5 - #1).

2.1 Supported Derivatives

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

Table 2-1. Supported Derivatives

Freescale Semiconductor	mpc5634m_bga208, mpc5634m_qfp144, mpc5634m_qfp176
-------------------------	---

All of the above microcontroller devices are collectively named as MPC5634M.

2.2 Overview

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

2.3 About this Manual

This Technical Reference employs the following typographical conventions:

Boldface type: Bold is used for important terms, notes and warnings.

Italic font: Italic typeface is used for code snippets in the text. Note that C language modifiers such "const" or "volatile" are sometimes omitted to improve readability of the presented code.

2.4 Acronyms and Definitions

Table 2-2. Acronyms and Definitions

Term	Definition
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSMI	Basic Software Make file Interface
CS	Chip Select
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
FIFO	First In First Out
MIDE	Multi Integrated Development Environment
MCU	Micro Controller Unit
LSB	Least Significant Bit
MSB	Most Significant Bit
RAM	Random Access Memory
SIU	Systems Integration Unit
SPI	Serial Peripheral Interface
SWS	Software Specification
VLE	Variable Length Encoding
XML	Extensible Markup Language
TSB	Timed Serial Bus
MSC	Micro Second Channel
N/A	Not Applicable

2.5 Reference List

Table 2-3. Reference List

#	Title	Version
2	MPC5634M Reference Manual	Rev. 6, 4 October 2011
2	AUTOSAR_SWS_SPI_HandlerDriver.pdf Reference Manual	V2.2.0 R3.0 Rev 0001

Chapter 3

Driver

3.1 Requirements

Requirements for this driver are detailed in the AUTOSAR SPI Driver software specification document. (See Table 2.5 - #1).

3.2 Driver Design Summary

The SPI Handler and Driver provide services for reading from and writing to devices connected via SPI busses. It provides access to SPI communication to several users (e.g., EEPROM, Watchdog, I/O ASICs). It also provides the required mechanism to configure the on-chip SPI peripheral.

This document describes the API, Mapping to SWS requirements for a monolithic SPI Handler and Driver. This software module includes handling and driving functionalities. Main objectives of this monolithic SPI Driver are to take the best of each microcontroller features and to allow implementation optimization depending on static configuration to fit as much as possible to ECU needs.

The general behavior of the SPI Handler and Driver could be asynchronous or synchronous according to the level of functionality selected.

The specification covers the Handler and Driver functionalities combined in one single module. The SPI handler controls multiple accesses to busses that could be located in the ECU Abstraction layer. The other part is the SPI driver that accesses the microcontroller hardware directly that could be located in the Microcontroller Abstraction layer.

SPI Dual Clock Mode

The SPI Driver allows to be used in Dual Clock Mode. This mode permits to change the clock reference(referred by the field SpiAlternateClockRef) and to keep the basic characteristics of the transmission(like baudrate). This is useful when it wants to be crossed to a low frequency(low power) or higher frequency.

Interrupt request usage

Every interrupt is guarded by #ifdef definitions that specify if the corresponding DSPI is used. If not, the interrupt function is removed. A template of the #ifdef guard is:

```
#if ((SPI_LEVEL_DELIVERED == LEVEL1) || (SPI_LEVEL_DELIVERED == LEVEL2))
```

```
#if (<DSPI_ENABLED> == STD_ON)
```

```
<ISR_function_name()>
```

```
#endif
```

```
#endif
```

One Job Sequences Optimization

This non-autosar feature activates the SPI transmission optimization for the sequences having only one job. At this moment, the optimization is in place only for synchronous transmissions. This option requires additional RAM for two internal buffers (for sequences having only one job, and for their linked channels), in order to cache configuration information to be used during transmission initialization. During Spi_Init() a caching action is performed on all sequences having only one job.

The cached values (sequence / job configuration parameters like parameters like Baud Rate, CS, Clock polarity) will be used in sync transmissions in order to optimize the time

Spi Optimized Seq Number: If, for a given configuration, the number of sequences having only one job exceeds this value, only first 'SpiOptimizedSeqNumber' sequences will operate in optimized transmission mode.

Spi Optimized Channels Number: The maximum number of channels in expected sequences with one job, targeted for optimization. If, for a given configuration, the number of channels in sequences having only one job exceeds this value, the sequences will be optimized only in the limit of 'SpiOptimizedChannelsNumber' consumption.

Multiple Synchronous Transmission

This non autosar feature allows simultaneous calls to Spi_SyncTransmit() for different threads. Two concurrent calls to Spi_SyncTransmit() will be allowed only if the related sequences do not share HW units. This feature was implemented from the need that different SPI buses may be requested concurrent according to BSW12151.

Description of the symbolic names

When the plugin is generated, symbolic names of the sequences, jobs and channels are created by define macros. The templates of the defines are:

- for sequences: `#define <SpiSequenceName> ((Spi_SequenceType)<SpiSequenceID>)`
- for jobs: `#define <SpiJobName> ((Spi_JobType)<SpiJobID>)`
- for channels: `#define <SpiChannelName> ((Spi_ChannelType)<SpiChannelID>)`

Name is the name of the container and the ID is configurable by the user

Spi Mixed Channel Size Optimization

Extension to AUTOSAR_SPI_SWS to allow mixing channels having different size (UINT16/UINT8) in the same configuration but also to have the driver optimized to save the RAM.

This feature is enabled at configuration time by checking the "SpiForceDataType" checkbox.

If it is unchecked the SPI driver behaves as is now: SPI_DataType is unit16.

If it is checked, the SPI_DataType is always uint8 and the Data buffers are accessed as uint8 or uint16 according to SpiDataWidth independently to Spi_DataType.

The data access will use following casting:

- uint8 for SpiDataWidth < 9
- uint16 for 9 <= SpiDataWidth < 17

3.3 Deviation from Requirements

Table 2.3 Deviations Status Column Description identifies the AUTOSAR requirements that are not fully implemented, implemented differently, or out of scope for the SPI driver. Table 2.4 SPI Deviations Table provides Status column description.

Table 3-1. Deviations Status Column Description

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

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

Table 3-2. SPI Deviations Table

Requirement	Status	Description	Notes
SPI088	N/S	In case of these two actions (job scheduling and end notification functions) are fully done by software; the order between them shall be first scheduling and after the call of end notification function. Otherwise, if they are done by hardware and the order could not be configured as required, the order shall be completely documented.	job and sequences notifications are performed before the scheduling of the next job (contrary to the recommendation given by SPI088) . In this way, calls like Spi_SetupIB() or Spi_WriteIB() can be targeted on the next schedulable jobs, before the starting of the job transfer.

3.4 Function Definitions

APIs of all functions supported by the driver are as per AUTOSAR SPI Driver software specification Version 3.0.

3.4.1 Function Spi_AsyncTransmit

This function triggers the asynchronous transmission for the given sequence.

Prototype: Std_ReturnType Spi_AsyncTransmit(Spi_SequenceType Sequence);

Table 3-3. Spi_AsyncTransmit Arguments

Type	Name	Direction	Description
Spi_SequenceType	Sequence	input	Sequence ID

Return: Std_ReturnType

- E_OK - Transmission command has been accepted
- E_NOT_OK - Transmission command has not been accepted

This function triggers the asynchronous transmission for the given sequence.

- Service ID: 0x03
- Sync/Async: Asynchronous
- Reentrancy: Reentrant

pre:: The driver needs to be initialized before calling `Spi_AsyncTransmit()` otherwise, the function `Spi_AsyncTransmit()` raises the development error if `SPI_DEV_ERROR_DETECT` is `STD_ON`. Pre-compile parameter `SPI_LEVEL_DELIVERED` shall be `LEVEL1` or `LEVEL2`.

3.4.2 Function Spi_Cancel

This function is used to request the cancelation of the given sequence.

Prototype: `void Spi_Cancel(Spi_SequenceType Sequence);`

Table 3-4. Spi_Cancel Arguments

Type	Name	Direction	Description
<code>Spi_SequenceType</code>	Sequence	input	Sequence ID

This function is used to request the cancelation of the given sequence.

- Service ID: 0x0c
- Sync/Async: Asynchronous
- Reentrancy: Reentrant

pre:: The driver needs to be initialized before calling `Spi_Cancel()` otherwise, the function `Spi_Cancel()` raises the development error if `SPI_DEV_ERROR_DETECT` is `STD_ON`. Pre-compile parameter `SPI_CANCEL_API` shall be `STD_ON`

post:: The SPI Handler/Driver is not responsible on external devices damages or undefined state due to cancelling a sequence transmission.

3.4.3 Function Spi_DeInit

This function de-initializes the SPI driver.

Prototype: `Std_ReturnType Spi_DeInit(void);`

Return: `Std_ReturnType`

- `E_OK` - de-initialisation command has been accepted
- `E_NOT_OK` - de-initialisation command has not been accepted

This function de-initializes the SPI driver using the pre-established configurations

- Service ID: 0x01

- Sync/Async: Synchronous
- Reentrancy: Non-Reentrant

pre:: The driver needs to be initialized before calling `Spi_DeInit()` otherwise, the function `Spi_DeInit()` raises the development error if `SPI_DEV_ERROR_DETECT` is `STD_ON`.

3.4.4 Function Spi_GetAsyncStatus

This function returns the status of the SPI driver related to async HW Units.

Prototype: `Spi_StatusType Spi_GetAsyncStatus(void);`

Return: `Spi_StatusType`

- `SPI_UNINIT` - The driver is un-initialized
- `SPI_IDLE` - The driver has no pending transfers
- `SPI_BUSY` - The driver is busy

Return `SPI_BUSY` if at least one async HW unit is busy.

pre:: Pre-compile parameter `SPI_LEVEL_DELIVERED` shall be `LEVEL2`.

3.4.5 Function Spi_GetHWUnitStatus

This function is used to request the status of a specific SPI peripheral unit.

Prototype: `Spi_StatusType Spi_GetHWUnitStatus(Spi_HWUnitType HWUnit);`

Table 3-5. Spi_GetHWUnitStatus Arguments

Type	Name	Direction	Description
<code>Spi_HWUnitType</code>	<code>HWUnit</code>	input	The HW peripheral for which we need the status

Return: `Spi_StatusType`

- `SPI_UNINIT` - The peripheral is un-initialized
- `SPI_IDLE` - The peripheral is in idle state
- `SPI_BUSY` - The peripheral is busy

This function is used to request the status of a specific SPI peripheral unit.

- Service ID: `0x0b`
- Sync/Async: Synchronous
- Reentrancy: Reentrant

pre:: The driver needs to be initialized before calling `Spi_GetHWUnitStatus()` otherwise, the function `Spi_GetHWUnitStatus()` raises the development error if `SPI_DEV_ERROR_DETECT` is `STD_ON`. `SPI_HW_STATUS_API == STD_ON`

3.4.6 Function Spi_GetJobResult

This function is used to request the status of a specific job.

Prototype: `Spi_JobResultType Spi_GetJobResult(Spi_JobType Job);`

Table 3-6. Spi_GetJobResult Arguments

Type	Name	Direction	Description
<code>Spi_JobType</code>	Job	input	Job ID

Return: `Spi_JobResultType`

- `SPI_JOB_OK` - The job ended successfully
- `SPI_JOB_PENDING` - The job is pending
- `SPI_JOB_FAILED` - The job has failed

This function is used to request the status of a specific job.

- Service ID: 0x07
- Sync/Async: Synchronous
- Reentrancy: Reentrant

pre:: The driver needs to be initialized before calling `Spi_GetJobResult()` otherwise, the function `Spi_GetJobResult()` raises the development error if `SPI_DEV_ERROR_DETECT` is `STD_ON`.

3.4.7 Function Spi_GetSequenceResult

This function is used to request the status of a specific sequence.

Prototype: `Spi_SeqResultType Spi_GetSequenceResult(Spi_SequenceType Sequence);`

Table 3-7. Spi_GetSequenceResult Arguments

Type	Name	Direction	Description
<code>Spi_SequenceType</code>	Sequence	input	Sequence ID

Return: `Spi_SeqResultType`

- `SPI_SEQ_OK` - The sequence ended successfully

- SPI_SEQ_PENDING - The sequence is pending
- SPI_SEQ_FAILED - The sequence has failed

This function is used to request the status of a specific sequence.

- Service ID: 0x08
- Sync/Async: Synchronous
- Reentrancy: Reentrant

pre:: The driver needs to be initialized before calling `Spi_GetSequenceResult()` otherwise, the function `Spi_GetSequenceResult()` raises the development error if `SPI_DEV_ERROR_DETECT` is `STD_ON`.

3.4.8 Function Spi_GetStatus

This function returns the status of the SPI driver.

Prototype: `Spi_StatusType Spi_GetStatus(void);`

Return: `Spi_StatusType`

- SPI_UNINIT - The driver is un-initialized
- SPI_IDLE - The driver has no pending transfers
- SPI_BUSY - The driver is busy

This function returns the status of the SPI driver.

- Service ID: 0x06
- Sync/Async: Synchronous
- Reentrancy: Reentrant

pre:: The driver needs to be initialized before calling `Spi_GetStatus()` otherwise, the function `Spi_GetStatus()` raises the development error if `SPI_DEV_ERROR_DETECT` is `STD_ON`.

3.4.9 Function Spi_GetVersionInfo

This function returns the version information for the SPI driver.

Prototype: `void Spi_GetVersionInfo(Std_VersionInfoType *VersionInfo);`

Table 3-8. Spi_GetVersionInfo Arguments

Type	Name	Direction	Description
Std_VersionInfoType *	VersionInfo	input, output	Pointer to where to store the version information of this module

Configuration structures.

This function returns the version information for the SPI driver.

- Service ID: 0x09
- Sync/Async: Synchronous
- Reentrancy: Non-Reentrant

pre.: Pre-compile parameter SPI_VERSION_INFO_API shall be STD_ON.

3.4.10 Function Spi_Init

This function initializes the SPI driver.

Prototype: void Spi_Init(const Spi_ConfigType *ConfigPtr);

Table 3-9. Spi_Init Arguments

Type	Name	Direction	Description
const Spi_ConfigType *	ConfigPtr	input	Specifies the pointer to the configuration set

This function initializes the SPI driver using the pre-established configurations

- Service ID: 0x00
- Sync/Async: Synchronous
- Reentrancy: Non-Reentrant

Note

Violates MISRA 2004 Advisory Rules 11.4

- See Spi_c_REF_3

3.4.11 Function Spi_MainFunction_Driving

This function shall asynchronously poll SPI interrupts and call ISR if appropriate.

Prototype: void Spi_MainFunction_Driving(void);

This function shall asynchronously poll SPI interrupts and call ISR if appropriate.

- Service ID: 0x11

pre:: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2.

3.4.12 Function Spi_ReadIB

This function reads the data from the buffer of a specific channel and puts at the specified memory location.

Prototype: Std_ReturnType Spi_ReadIB(Spi_ChannelType Channel, Spi_DataType *DataBufferPtr);

Table 3-10. Spi_ReadIB Arguments

Type	Name	Direction	Description
Spi_ChannelType	Channel	input	Channel ID
Spi_DataType *	DataBufferPtr	input, output	Pointer to the memory location that will be written with the data in the internal buffer

Return: Std_ReturnType

- E_OK - read command has been accepted
- E_NOT_OK - read command has not been accepted

This function reads the data from the buffer of a specific channel and puts at the specified memory location.

- Service ID: 0x04
- Sync/Async: Synchronous
- Reentrancy: Reentrant

pre:: The driver needs to be initialized before calling Spi_ReadIB() otherwise, the function Spi_ReadIB() raises the development error if SPI_DEV_ERROR_DETECT is STD_ON. Pre-compile parameter SPI_CHANNEL_BUFFERS_ALLOWED shall be USAGE0 or USAGE2.

3.4.13 Function Spi_SetAsyncMode

This function specifies the asynchronous mode for the SPI busses handled asynchronously.

Prototype: Std_ReturnType Spi_SetAsyncMode(Spi_AsyncModeType AsyncMode);

Table 3-11. Spi_SetAsyncMode Arguments

Type	Name	Direction	Description
Spi_AsyncModeType	AsyncMode	input	This parameter specifies the asynchronous operating mode (SPI_POLLING_MODE or SPI_INTERRUPT_MODE)

Return: Std_ReturnType

- E_OK - The command ended successfully
- E_NOT_OK - The command has failed

This function specifies the asynchronous mode for the SPI busses handled asynchronously.

- Service ID: 0x0d
- Sync/Async: Synchronous
- Reentrancy: Non-Reentrant

pre:: The driver needs to be initialized before calling Spi_SetAsyncMode() otherwise, the function Spi_SetAsyncMode() raises the development error if SPI_DEV_ERROR_DETECT is STD_ON. Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL2

3.4.14 Function Spi_SetupEB

This function setup an external buffer to be used by a specific channel.

Prototype: Std_ReturnType Spi_SetupEB(Spi_ChannelType Channel, const Spi_DataType *SrcDataBufferPtr, Spi_DataType *DesDataBufferPtr, Spi_NumberOfDataType Length);

Table 3-12. Spi_SetupEB Arguments

Type	Name	Direction	Description
Spi_ChannelType	Channel	input	Channel ID
const Spi_DataType *	SrcDataBufferPtr	input	Pointer to the memory location that will hold the transmitted data
Spi_NumberOfDataType	Length	input	Length of the data in the external buffer
Spi_DataType *	DesDataBufferPtr	output	Pointer to the memory location that will hold the received data

Return: Std_ReturnType

- E_OK - Setup command has been accepted
- E_NOT_OK - Setup command has not been accepted

This function setup an external buffer to be used by a specific channel.

- Service ID: 0x05
- Sync/Async: Synchronous
- Reentrancy: Reentrant

pre:: The driver needs to be initialized before calling `Spi_SetupEB()` otherwise, the function `Spi_SetupEB()` raises the development error if `SPI_DEV_ERROR_DETECT` is `STD_ON`. Pre-compile parameter `SPI_CHANNEL_BUFFERS_ALLOWED` shall be `USAGE1` or `USAGE2`.

Note

Violates MISRA 2004 Advisory Rules 11.4

- See `Spi_c_REF_3`

3.4.15 Function Spi_SyncTransmit

This function is used for synchronous transmission of a given sequence.

Prototype: `Std_ReturnType Spi_SyncTransmit(Spi_SequenceType Sequence);`

Table 3-13. Spi_SyncTransmit Arguments

Type	Name	Direction	Description
<code>Spi_SequenceType</code>	Sequence	input	Sequence ID

Return: `Std_ReturnType`

- `E_OK` - Transmission command has been completed successfully
- `E_NOT_OK` - Transmission command has not been accepted

This function is used for synchronous transmission of a given sequence.

- Service ID: 0x0a
- Sync/Async: Synchronous
- Reentrancy: Reentrant

pre:: The driver needs to be initialized before calling `Spi_SyncTransmit()`. otherwise, the function `Spi_SyncTransmit()` raises the development error if `SPI_DEV_ERROR_DETECT` is `STD_ON`. Pre-compile parameter `SPI_LEVEL_DELIVERED` shall be `LEVEL0` or `LEVEL2`

3.4.16 Function Spi_WriteIB

This function writes the given data into the buffer of a specific channel.

Prototype: Std_ReturnType Spi_WriteIB(Spi_ChannelType Channel, const Spi_DataType *DataBufferPtr);

Table 3-14. Spi_WriteIB Arguments

Type	Name	Direction	Description
Spi_ChannelType	Channel	input	Channel ID
const Spi_DataType *	DataBufferPtr	input	Pointer to source data buffer

Return: Std_ReturnType

- E_OK - Command has been accepted
- E_NOT_OK - Command has not been accepted

This function writes the given data into the buffer of a specific channel.

- Service ID: 0x02
- Sync/Async: Synchronous
- Reentrancy: Reentrant

pre:: The driver needs to be initialized before calling Spi_WriteIB() otherwise, the function Spi_WriteIB() raises the development error if SPI_DEV_ERROR_DETECT is STD_ON. Pre-compile parameter SPI_CHANNEL_BUFFERS_ALLOWED shall be USAGE0 or USAGE2.

3.4.17 Function Spi_SetHWUnitAsyncMode

This function specifies the asynchronous mode for a given HWUnit.

Prototype: Std_ReturnType Spi_SetHWUnitAsyncMode(Spi_HWUnitType HWUnit, Spi_AsyncModeType AsyncMode);

Table 3-15. Spi_SetHWUnitAsyncMode Arguments

Type	Name	Direction	Description
Spi_HWUnitType	HWUnit	input	The ID of the HWUnit to be configured
Spi_AsyncModeType	AsyncMode	input	This parameter specifies the asynchronous operating mode (SPI_POLLING_MODE or SPI_INTERRUPT_MODE)

Return: Std_ReturnType

- E_OK - The command ended successfully
- E_NOT_OK - The command has failed

This function specifies the asynchronous mode for the SPI busses handled asynchronously. For synchronous HW units, the function has no impact. The function will fail in two cases:

- driver not initialised (SPI_E_UNINIT reported by DET)
- a sequence transmission is pending the the asynchronous HW unit (SPI_E_SEQ_PENDING reported by DET)

pre:: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL2 and SPI_HWUNIT_ASYNC_MODE should be on STD_ON

3.4.18 Function Spi_SetClockMode

This function shall set different MCU clock configuration .

Prototype: Std_ReturnType Spi_SetClockMode(Spi_DualClockMode ClockMode);

Table 3-16. Spi_SetClockMode Arguments

Type	Name	Direction	Description
Spi_DualClockMode	ClockMode	input	Clock mode to be set (SPI_NORMAL SPI_ALTERNATE).

Return: Std_ReturnType

- E_OK - The driver is initialised and in an IDLE state. The clock mode can be changed.
- E_NOT_OK - The driver is NOT initialised OR is NOT in an IDLE state. The clock mode can NOT be changed.

This function shall set different MCU clock configuration .

pre:: Pre-compile parameter SPI_DUAL_CLOCK_MODE shall be STD_ON.

3.4.19 Function Spi_TSB_AS DR_DataUpdate

This function shall to enable user to write data in AS DR (alternate serialisation register) while using Micro Second Bus functionality .

Prototype: Std_ReturnType Spi_TSB_AS DR_DataUpdate(Spi_JobType TSBJob, uint32 AS DR_Data);

Table 3-17. Spi_WriteIB Arguments

Type	Name	Direction	Description
Spi_JobType	TSBJob	input	Job ID
uint32	ASDR_Data	input	Data to write in ASDR (alternate serialisation register)

Return: Std_ReturnType

- E_OK - Command has been accepted
- E_NOT_OK - Command has not been accepted

This function shall to enable user to write data in ASDR (alternate serialisation register) while using Micro Second Bus functionality .

- Service ID: 0x82

pre:: The driver needs to be initialized before calling Spi_TSB_ASDR_DataUpdate() otherwise, the function Spi_TSB_ASDR_DataUpdate() raises the development error if SPI_DEV_ERROR_DETECT is STD_ON. Pre-compile parameter SPI_TSB_MODE shall be STD_ON

3.4.20 Function Spi_TSBStart

This function shall to enable user to access the Micro Second Bus functionality

Prototype: Std_ReturnType Spi_TSBStart(Spi_JobType TSBJob);

Table 3-18. Spi_WriteIB Arguments

Type	Name	Direction	Description
Spi_JobType	TSBJob	input	Job ID

Return: Std_ReturnType

- E_OK - Command has been accepted
- E_NOT_OK - Command has not been accepted

This function enable the user to access the Micro Second Bus functionality.

- Service ID: 0x82

pre:: The driver needs to be initialized before calling Spi_TSBStart() otherwise, the function Spi_TSBStart() raises the development error if SPI_DEV_ERROR_DETECT is STD_ON. Pre-compile parameter SPI_TSB_MODE shall be STD_ON

3.4.21 Function Spi_TSBStop

This function shall to stop the transfer in TSB mode.

Prototype: Std_ReturnType Spi_TSBStop(Spi_JobType TSBJob);

Table 3-19. Spi_WriteIB Arguments

Type	Name	Direction	Description
Spi_JobType	TSBJob	input	Job ID

Return: Std_ReturnType

- E_OK - Command has been accepted
- E_NOT_OK - Command has not been accepted

This function shall to stop the transfer in TSB mode.

- Service ID: 0x82

pre:• The driver needs to be initialized before calling Spi_TSBStop() otherwise, the function Spi_TSBStop() raises the development error if SPI_DEV_ERROR_DETECT is STD_ON.Pre-compile parameter SPI_TSB_MODE shall be STD_ON

3.4.22 Function Index

Table 3-20. Quick Function Reference

Type	Name	Arguments
Std_ReturnType	Spi_SetClockMode	Spi_DualClockMode ClockMode
Std_ReturnType	Spi_AsyncTransmit	Spi_SequenceType Sequence
void	Spi_Cancel	Spi_SequenceType Sequence
Std_ReturnType	Spi_DeInit	void
Spi_StatusType	Spi_GetAsyncStatus	void
Spi_StatusType	Spi_GetHWUnitStatus	Spi_HWUnitType HWUnit
Spi_JobResultType	Spi_GetJobResult	Spi_JobType Job
Spi_SeqResultType	Spi_GetSequenceResult	Spi_SequenceType Sequence
Spi_StatusType	Spi_GetStatus	void
void	Spi_GetVersionInfo	Std_VersionInfoType * VersionInfo
void	Spi_Init	const Spi_ConfigType * ConfigPtr

Table continues on the next page...

Table 3-20. Quick Function Reference (continued)

Type	Name	Arguments
void	Spi_MainFunction_Driving	void
Std_ReturnType	Spi_ReadlB	Spi_ChannelType Channel Spi_DataType * DataBufferPtr
Std_ReturnType	Spi_SetAsyncMode	Spi_AsyncModeType AsyncMode
Std_ReturnType	Spi_SetHWUnitAsyncMode	Spi_HWUnitType HWUnit Spi_AsyncModeType AsyncMode
Std_ReturnType	Spi_SetupEB	Spi_ChannelType Channel const Spi_DataType * SrcDataBufferPtr Spi_NumberOfDataType Length Spi_DataType * DesDataBufferPtr
Std_ReturnType	Spi_SyncTransmit	Spi_SequenceType Sequence
Std_ReturnType	Spi_WritelB	Spi_ChannelType Channel const Spi_DataType * DataBufferPtr
Std_ReturnType	Spi_TSBStart	Spi_JobType TSBJob
Std_ReturnType	Spi_TSBStop	Spi_JobType TSBJob
Std_ReturnType	Spi_TSB_ASDR_DataUpdate	Spi_JobType TSBJob, uint32 ASDR_Data

3.5 Enum Definitions

APIs of all enumerations supported by the driver are as per AUTOSAR SPI Driver software specification Version 3.0.

3.5.1 Enumeration Spi_DualClockMode

This enum specifies the modes of the clock.

This enum specifies the modes of the clock:

Table 3-21. Enumeration Spi_DualClockMode Values

Value	Description
SPI_NORMAL = 0	Clock reference is from SpiClockRef.
SPI_ALTERNATE	Clock reference is from SpiAlternateClockRef.

3.5.2 Enumeration Spi_JobResultType

This type defines a range of specific Jobs status for SPI handler/Driver.

Table 3-22. Enumeration Spi_JobResultType Values

Value	Description
SPI_JOB_OK = 0	The last transmission of the Job has been finished successfully.
SPI_JOB_PENDING	The SPI handler/Driver is performing a SPI Job.
SPI_JOB_FAILED	The last transmission of the Job has failed.

3.5.3 Enumeration Spi_SeqResultType

This type defines a range of specific Sequences status for SPI handler/Driver.

Table 3-23. Enumeration Spi_SeqResultType Values

Value	Description
SPI_SEQ_OK = 0	The last transmission of the Sequence has been finished successfully.
SPI_SEQ_PENDING	The SPI handler/Driver is performing a SPI Sequence.
SPI_SEQ_FAILED	The last transmission of the Sequence has failed.
SPI_SEQ_CANCELLED	The last transmission of the Sequence has been cancelled by the user.

3.5.4 Enumeration Spi_StatusType

This type defines a range of specific status for SPI handler/Driver.

Table 3-24. Enumeration Spi_StatusType Values

Value	Description
SPI_UNINIT = 0	Not initialized or not usable.
SPI_IDLE	Not currently transmitting any jobs.
SPI_BUSY	Is performing a SPI Job(transmit).

3.5.5 Enumeration Spi_AsyncModeType

This type defines a range of Async mode for SPI handler/Driver.

Table 3-25. Enumeration Spi_StatusType Values

Value	Description
SPI_POLLING_MODE = 0	The asynchronous mechanism is ensured by polling, so interrupts related to SPI busses handled asynchronously are disabled.
SPI_INTERRUPT_MODE	The asynchronous mechanism is ensured by interrupt, so interrupts related to SPI busses handled asynchronously are enabled.

3.6 Structure Definitions

APIs of all structures supported by the driver are as per AUTOSAR SPI Driver software specification Version 3.0.

3.6.1 Structure Spi_HWUnitConfig

This structure holds the HWUnit configuration parameters.

Declaration

```
typedef struct
{
    uint8 Offset,
    uint32 IsSync,
    uint16 UseDma,
    uint8 TxDmaChannel,
    uint8 TxDmaChannelAux,
    uint8 RxDmaChannel
} Spi_HWUnitConfig;
```

Table 3-26. Structure Spi_HWUnitConfig member description

Member	Description
Offset	DSPI HWunit physical offset on SOC.
IsSync	indicates if the HW unit is configured as Sync or Async
UseDma	boolean flag indicating if DMA will be used or not for this DSPI unit
TxDmaChannel	Master TX DMA channel - enabled by the DSPI TX source.
TxDmaChannelAux	Auxiliary TX DMA channel - triggered by the master TX Dma.
RxDmaChannel	RX DMA channel - enabled by the DSPI RX source.

3.6.2 Structure Spi_ChannelConfig

The structure keeps a channel configuration.

Declaration

```
typedef struct
{
    Spi_BufferType BufferType,
    Spi_DataType DefaultTransmitValue,
    Spi_NumberOfDataType Length,
    const Spi_BufferDescriptor * BufferDescriptor,
    Spi_ChannelState * ChannelState
} Spi_ChannelConfig;
```

Table 3-27. Structure Spi_ChannelConfig member description

Member	Description
BufferType	Buffer Type IB/EB.
DefaultTransmitValue	Default Transmit Value.
Length	Data length.
BufferDescriptor	Buffer Descriptor.
ChannelState	Implementation specific field referencing the channel internal state.

3.6.3 Structure Spi_JobConfig

The structure keeps a job configuration.

Declaration

```
typedef struct
{
    Spi_ChannelType NumChannels,
    const Spi_ChannelType * ChannelIndexList,
    Spi_NotifyType * EndNotification,
    Spi_NotifyType * StartNotification,
    sint8 Priority,
    Spi_JobState * JobState,
    Spi_HWUnitType HWUnit,
    uint32 HWoffset,
    Spi_ExternalDeviceType ExternalDevice,
    Spi_LLD_DeviceAttributesConfig ExternalDeviceAttrs
} Spi_JobConfig;
```

Table 3-28. Structure Spi_JobConfig member description

Member	Description
NumChannels	Number of channels in the job.
ChannelIndexList	Channel index list.
EndNotification	Job notification.
StartNotification	Job Start Notification

Table continues on the next page...

Table 3-28. Structure Spi_JobConfig member description (continued)

Member	Description
Priority	Priority.
JobState	Implementation specific field referencing the channel internal state.
HWUnit	HWUnit.
HWOffset	HW Unit offset.
ExternalDevice	ExternalDevice.
ExternalDeviceAttrs	Implementation specific field: cached LLD device attributes.

3.6.4 Structure Spi_SequenceConfig

The structure keeps a sequence configuration.

Declaration

```
typedef struct
{
    Spi_JobType NumJobs,
    const Spi_JobType * JobIndexList,
    Spi_NotifyType * EndNotification,
    uint8 Interruptible
} Spi_SequenceConfig;
```

Table 3-29. Structure Spi_SequenceConfig member description

Member	Description
NumJobs	Number of jobs in the sequence.
JobIndexList	Job index list.
EndNotification	Job notification handler.
Interruptible	Boolean indicating if the Sequence is interruptible or not.

3.6.5 Structure Spi_TSBConfig

The structure keeps a sequence configuration.

Declaration

```
typedef struct
{
    Spi_JobType Spi_TSB_Job,
    Spi_LLD_TSBAttributesConfig ExternalDeviceTSBAttrs
} Spi_TSBConfig;
```

Table 3-30. Structure Spi_SequenceConfig member description

Member	Description
Spi_TSB_Job	JobId of the configured TSB job.
ExternalDeviceTSBAttrs	brief This structure holds the TSB mode selection on HWUnits.

3.7 Define Definitions

APIs of all defines supported by the driver are as per AUTOSAR SPI Driver software specification Version 3.0.

3.7.1 Define SPI_ASYNCTRANSMIT_ID

API service ID for SPI async transmit function.

Definition: `#define SPI_ASYNCTRANSMIT_ID (uint8) 0x03u`

3.7.2 Define SPI_CANCEL_ID

API service ID for SPI cancel function.

Definition: `#define SPI_CANCEL_ID (uint8) 0x0Cu`

3.7.3 Define SPI_DEINIT_ID

API service ID for SPI DeInit function.

Definition: `#define SPI_DEINIT_ID (uint8) 0x01u`

3.7.4 Define SPI_GETHWUNITSTATUS_ID

API service ID for SPI get hwunit status function.

Definition: `#define SPI_GETHWUNITSTATUS_ID (uint8) 0x0Bu`

3.7.5 Define SPI_GETJOBRESULT_ID

API service ID for SPI get job result function.

Definition: `#define SPI_GETJOBRESULT_ID (uint8) 0x07u`

3.7.6 Define SPI_GETSEQUENCERESULT_ID

API service ID for SPI get sequence result function.

Definition: `#define SPI_GETSEQUENCERESULT_ID (uint8) 0x08u`

3.7.7 Define SPI_GETSTATUS_ID

API service ID for SPI get status function.

Definition: `#define SPI_GETSTATUS_ID (uint8) 0x06u`

3.7.8 Define SPI_GETVERSIONINFO_ID

API service ID for SPI get version info function.

Definition: `#define SPI_GETVERSIONINFO_ID (uint8) 0x09u`

3.7.9 Define SPI_INIT_ID

API service ID for SPI Init function.

Definition: `#define SPI_INIT_ID (uint8) 0x00u`

3.7.10 Define SPI_MAINFUNCTION_DRIVING_ID

API service ID for SPI main function.

Definition: `#define SPI_MAINFUNCTION_DRIVING_ID (uint8) 0x11u`

3.7.11 Define SPI_MODULE_ID

Parameters that shall be published within the Spi driver header file and also in the module's description file.

Definition:`#define SPI_MODULE_ID 83`

3.7.12 Define SPI_READIB_ID

API service ID for SPI read IB function.

Definition:`#define SPI_READIB_ID (uint8) 0x04u`

3.7.13 Define SPI_SETASYNCMODE_ID

API service ID for SPI set async mode function.

Definition:`#define SPI_SETASYNCMODE_ID (uint8) 0x0Du`

3.7.14 Define SPI_SETCLOCKMODE_ID

API service ID for SPI Set Clock Mode.

Definition:`#define SPI_SETCLOCKMODE_ID (uint8)0x81u`

3.7.15 Define SPI_SETHWUNITASYNCMODE_ID

API service ID for SPI set HW Unit async mode.

Definition:`#define SPI_SETHWUNITASYNCMODE_ID (uint8)0x80u`

3.7.16 Define SPI_SETUPEB_ID

API service ID for SPI setup EB function.

Definition:`#define SPI_SETUPEB_ID (uint8) 0x05u`

3.7.17 Define SPI_SYNCTRANSMIT_ID

API service ID for SPI sync transmit function.

Definition: `#define SPI_SYNCTRANSMIT_ID (uint8) 0x0Au`

3.7.18 Define SPI_VENDOR_ID

Parameters that shall be published within the Spi driver header file and also in the module's description file.

Definition: `#define SPI_VENDOR_ID 43`

3.7.19 Define SPI_WRITEIB_ID

API service ID for SPI write IB function.

Definition: `#define SPI_WRITEIB_ID (uint8) 0x02u`

3.7.20 Define SPI_E_ALREADY_INITIALIZED

API SPI_Init service called while the SPI driver has already been intialized.

Definition: `#define SPI_E_ALREADY_INITIALIZED (uint8) 0x4Au`

3.7.21 Define SPI_E_CONFIG_OUT_OF_RANGE

The number of sequences, jobs or channels in the configuration exceeds precompile time related sizes: SPI_MAX_SEQUENCE, SPI_MAX_JOB or SPI_MAX_CHANNEL.

Definition: `#define SPI_E_CONFIG_OUT_OF_RANGE (uint8) 0x5Au`

3.7.22 Define SPI_E_JOB_EMPTY

No channel in job.

Definition: `#define SPI_E_JOB_EMPTY (uint8) 0x5Du`

3.7.23 Define SPI_E_PARAM_CHANNEL

API service called with wrong parameter.

Definition: `#define SPI_E_PARAM_CHANNEL (uint8)0x0Au`

3.7.24 Define SPI_E_PARAM_EB_UNIT

When a sequence contains uninitialized external buffers.

Definition: `#define SPI_E_PARAM_EB_UNIT (uint8)0x5Bu`

3.7.25 Define SPI_E_PARAM_JOB

API service called with wrong parameter.

Definition: `#define SPI_E_PARAM_JOB (uint8)0x0Bu`

3.7.26 Define SPI_E_PARAM_LENGTH

API service called with wrong parameter.

Definition: `#define SPI_E_PARAM_LENGTH (uint8)0x0Du`

3.7.27 Define SPI_E_PARAM_SEQ

API service called with wrong parameter.

Definition: `#define SPI_E_PARAM_SEQ (uint8)0x0Cu`

3.7.28 Define SPI_E_PARAM_UNIT

API service called with wrong parameter.

Definition: `#define SPI_E_PARAM_UNIT (uint8)0x0Eu`

3.7.29 Define SPI_E_SEQ_EMPTY

No job in sequence.

Definition: `#define SPI_E_SEQ_EMPTY (uint8) 0x5Cu`

3.7.30 Define SPI_E_SEQ_IN_PROCESS

Synchronous transmission service called at wrong time.

Definition: `#define SPI_E_SEQ_IN_PROCESS (uint8) 0x3Au`

3.7.31 Define SPI_E_SEQ_PENDING

Services called in a wrong sequence.

Definition: `#define SPI_E_SEQ_PENDING (uint8) 0x2Au`

3.7.32 Define SPI_E_UNINIT

API service used without module initialization.

Definition: `#define SPI_E_UNINIT (uint8) 0x1Au`

3.7.33 Define SPI_SETTSBMODE_ID

API service ID for SPI settsbmode function.

Definition: `#define SPI_SETTSBMODE_ID (uint8) 0x82u`

3.8 Symbolic Names DISCLAIMER

All containers having the symbolic name tag set as true in the Autosar schema will generate defines like:

```
#define <Container_Short_Name> <Container_ID>
```

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

3.9 Configuration Parameters

As per the AUTOSAR specification the driver has two types of configurations parameters – Pre-Compile parameters and Post-Build parameters. Pre-Compile parameters are stored in the file “Spi_Cfg.h” & “Spi_Cfg.c”. The Post-Build parameters are stored in the file “Spi_PBcfg.c”.

The files to be used for different configuration types are listed below.

- 1) Variant PC: Spi_Cfg.h, Spi_Cfg.c
- 2) Variant PB: Spi_Cfg.h, Spi_PBcfg.c
- 3) Variant LT: Spi_Cfg.h, Spi_Lcfg.c


A section for Spi_PBcfg.c file is needed in linker file to place the post build configuration in desired location. Please refer to section-8 of "AUTOSAR_SWS_C_ImplementationRules.pdf" for complete details on configuration types.

3.9.1 Pre-Compile Parameters




Pre-Compile parameters, their possible values and meaning are described in the following text. Pre-Compile parameters are implemented as preprocessor defines.

The configuration screens for spi precompile parameters in Tresos® Studio configuration tool from Tresos Tresos 2010a.sr4 20100415-release2010a-sr4 is given below:


Spi


























Name  Spi

General SpiDriver SpiPhyUnit Published Information

Config Variant  VariantLinkTime  

▼ SpiGeneral


 SpiGeneral

SpiCancelApi	 <input checked="" type="checkbox"/> 	
SpiChannelBuffersAllowed (0 -> 2)	 2 	
SpiDevErrorDetect	 <input checked="" type="checkbox"/> 	SpiHwStatusApi  <input checked="" type="checkbox"/> 
SpiInterruptibleSeqAllowed	 <input type="checkbox"/> 	
SpiLevelDelivered (0 -> 2)	 2 	
SpiVersionInfoApi	 <input checked="" type="checkbox"/> 	
SpiClockRef	 /Mcu/Mcu/Mcu_Config_PB_0/McuClockSettingConfig/Mcu	
SpiGlobalDmaEnable	 <input checked="" type="checkbox"/> 	
SpiSyncTransmitTimeout (1 -> 65535)	 10000 	
SpiOptimizeOneJobSequences	 <input checked="" type="checkbox"/> 	
SpiOptimizedSeqNumber (0 -> 64)	 0 	
SpiOptimizedChannelsNumber (0 -> 64)	 0 	

► SpiNonAUTOSAR

Figure 3-1. SpiGeneral Configuration

▼ SpiNonAUTOSAR

 SpiNonAUTOSAR














SpiEnableMultiSyncTransmit*	 <input checked="" type="checkbox"/> 	SpiAllowBigSizeCollections*	 <input checked="" type="checkbox"/> 
SpiEnableHWUnitAsyncMode*	 <input checked="" type="checkbox"/> 	SpiEnableDualClockMode*	 <input checked="" type="checkbox"/> 
SpiJobStartNotificationenable	 <input type="checkbox"/> 	SpiForceDataType	 <input type="checkbox"/> 
SpiAlternateClockRef*	 /Mcu/Mcu/McuModuleConfiguration_0/McuClock!		

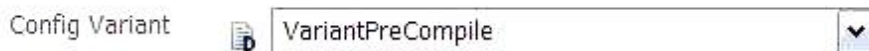
Figure 3-2. SpiNonAutosar Configuration

Note

If pre-compile variant is preferred,

- IMPLEMENTATION_CONFIG_VARIANT in Tresos GUI should be selected as “VariantPreCompile” as shown below.

-



- Spi_Cfg.c should be compiled
- Spi_PBcfg.c should not be compiled

3.9.1.1 DualReceiverSupport

Table 3-31. DualReceiverSupport

Description	A feature to enable to switch the set of PCS signals that are driven during the first part of the frame to a different set of PCS signals during the second part of the frame.
Class	Non-Autosar Parameter
Range	true,false
Default	false
Source File	Spi_PbCfg.c
Source Representation	NA

3.9.1.2 SpiCancelApi

Table 3-32. SpiCancelApi

Description	Defines whether the function Spi_Cancel shall be available (STD_ON) or not (STD_OFF) at compile time.
Class	Autosar Parameter
Range	True, false
Default	true
Source File	Spi_Cfg.h
Source Representation	#define SPI_CANCEL_API <STD_OFF, STD_ON>

3.9.1.3 SpiChannelBuffersAllowed

Table 3-33. SpiChannelBuffersAllowed

Description	USAGE0: Spi handler/driver manages only Internal Buffers USAGE1: Spi handler/driver manages only External Buffers USAGE2: Spi handler/driver manages both Buffers types.
Class	Autosar Parameter
Range	USAGE0, USAGE1 or USAGE2
Default	USAGE0
Source File	Spi_Cfg.h
Source Representation	#define SPI_CHANNEL_BUFFERS_ALLOWED<USAGE0, USAGE1, USAGE2>

3.9.1.4 SpiDevErrorDetect

Table 3-34. SpiDevErrorDetect

Description	Defines whether support of development error detection should be included at compile time (STD_ON) or excluded (STD_OFF). When enabled API parameter checking is done at runtime.
Class	Autosar Parameter
Range	True,false
Default	false
Source File	Spi_Cfg.h
Source Representation	#define Spi_DEV_ERROR_DETECT<STD_OFF, STD_ON>

3.9.1.5 SpiHWStatusAPI

Table 3-35. SpiHWStatusAPI

Description	Defines whether function “Spi_GetHWUnitStatus” is available (STD_ON) or unavailable (STD_OFF) at compile time.
Class	Autosar Parameter
Range	True, false
Default	true
Source File	Spi_Cfg.h
Source Representation	#define SPI_HW_STATUS_API<STD_OFF, STD_ON>

3.9.1.6 SpiInterruptibleSeqAllowed

Table 3-36. SpiInterruptibleSeqAllowed

Description	Defines whether a sequence should be interruptible at compile time (STD_ON) or excluded (STD_OFF).
Class	Autosar Parameter
Range	True, false
Default	true
Source File	Spi_Cfg.h
Source Representation	#define SPI_INTERRUPTIBLE_SEQ_ALLOWED<STD_OFF, STD_ON>

3.9.1.7 SpiLevelDelivered

Table 3-37. SpiLevelDelivered

Description	Defines the type of communication that the Spi handler/driver will use. LEVEL0: Only Simple synchronous communication is allowed. LEVEL1: Basic Asynchronous communication is allowed. A priority policy is applied using this configuration level. LEVEL2: Using this level the Spi communication will be based on asynchronous behavior or synchronous handling. Using this level an interrupt or polling mechanism can be used.
Class	Autosar Parameter
Range	LEVEL0, LEVEL1 & LEVEL2
Default	LEVEL2
Source File	Spi_Cfg.h
Source Representation	#define SPI_LEVEL_DELIVERED< LEVEL0, LEVEL1, LEVEL2>

Note

SPI_LEVEL_0 is sufficient when Synchronous transmission is used. Spi_Irq.c is not needed if only synchronous transmission is used. The DMA functionality is not used when level 0 is selected. The DMA can be used only in level 1 or 2 and when the SpiPhyUnitAsyncMethod is set to DMA.

3.9.1.8 SpiLevelDelivered

Table 3-38. SpiLevelDelivered

Description	Defines whether driver function will be included at compile time (STD_ON) or excluded (STD_OFF).
Class	Autosar Parameter
Range	True, false

Table continues on the next page...

Table 3-38. SpiLevelDelivered (continued)

Default	true
Source File	Spi_Cfg.h
Source Representation	#define SPI_VERSION_INFO_API<STD_OFF, STD_ON>

3.9.1.9 SpiClockReference

Table 3-39. SpiClockReference

Description	The reference to MCU ClockReference Point to get the SPI clock
Class	Implementation specific parameter
Range	N/A
Default	N/A
Source File	Spi_Cfg.h
Source Representation	It is only used in code template.

3.9.1.10 SpiGlobalDmaEnable

Table 3-40. SpiGlobalDmaEnable

Description	<p>This parameter is used only for Asynchronous Transmission.</p> <p>If enabled, the DSPI units working in asynchronous transferring mode may be configured in DMA operation mode or in PIO_FIFO mode according to the value of the SpiPhyUnitAsyncMethod field.</p> <p>If disabled, all transfers will be made in PIO_FIFO mode.</p> <p>DMA - DMA is used as transfer method for transferring data from channel buffer to DSPI Transmit FIFO register and receives data from DSPI Receive FIFO register to channel buffer.</p> <p>PIO_FIFO - Programmed Input Output First in First Out – The Spi driver transfers a maximum of four data entries from channel buffer to DSPI Transmit FIFO registers and receives a maximum of four data entries from DSPI Receive FIFO registers to channel buffer.</p>
Class	Implementation specific parameter
Range	true, false
Default	true
Source File	Spi_Cfg.h
Source Representation	#define SPI_DMA_USED <STD_OFF, STD_ON>
Reference	Refer chapter 5.10 of the Integration Manual

3.9.1.11 SpiSyncTransmitTimeout

Table 3-41. SpiSyncTransmitTimeout

Description	Timeout value (in microseconds) used by SPI_SyncTransmit function to wait for TX/RX FIFO to be emptied/filled. When this timeout is elapsed, an error is reported to the Dem module
Class	Implementation specific parameter
Range	1-65535
Default	50000
Source File	Spi_Cfg.h
Source Representation	#define SPI_TIMEOUT_COUNTER <number_of_waiting_loops>

Note

The precision of this value is quite low, as the time expressed in microseconds is converted in number of waiting loops and the duration of a loop cycle depends of compiler configuration. You must configure this timeout at a value much larger than the maximum transmission latency used in the configuration. Ex: Supposing that the slowest transmission is for a device accepting frames of 16 bits at a baud rate of 500KHz, and demanding one Clk before the CS activation and after CS releasing - Frame transmissions duration will be: $2\mu s T_{asc} + 2\mu s \times 16 + 2\mu s T_{pcssck} = 36\mu s$ - You may configure SpiSyncTransmitTimeout at $360\mu s$ (10 times the maximum latency duration)

3.9.1.12 SpiOptimizeOneJobSequences

Table 3-42. SpiOptimizeOneJobSequences

Description	Activates the SPI transmission optimization for the sequences having only one job. At his moment, the optimization is in place only for synchronous transmissions.
Class	Implementation specific parameter
Range	true, false
Default	false
Source File	Spi_Cfg.h
Source Representation	#define SPI_OPTIMIZE_ONE_JOB_SEQUENCES (<STD_OFF, STD_ON>)

Note

This option requires additional RAM for two internal buffers (for sequences having only one job, and for their linked channels), in order to cache configuration information to be used during transmission initialization. During Spi_Init() a caching action is performed on all sequences having only one job

3.9.1.13 SpiOptimizedChannelsNumber**Table 3-43. SpiOptimizedChannelsNumber**

Description	The maximum number of channels in expected sequences with one job, targeted for optimization. If, for a given configuration, the number of channels in sequences having only one job exceeds this value, the sequences will be optimized only in the limit of 'SpiOptimizedChannelsNumber' consumption.
Class	Implementation specific parameter
Range	0-64
Default	0 (the total number of channels in one job sequences in configuration)
Source File	Spi_Cfg.h
Source Representation	#define SPI_OPTIMIZED_CHANNEL_BUFFER_SIZE (<Channel_Count>)

Note

A value of 0 will define the buffer size for fitting all optimize-able sequences

3.9.1.14 SpiOptimizedSeqNumber**Table 3-44. SpiOptimizedSeqNumber**

Description	The maximum number of expected sequences with one job, targeted for optimization. If, for a given configuration, the number of sequences having only one job exceeds this value, only first 'SpiOptimizedSeqNumber' sequences will operate in optimized transmission mode.
Class	Implementation specific parameter
Range	0-64
Default	0 (the total number of one job sequences in configuration)
Source File	Spi_Cfg.h
Source Representation	#define SPI_OPTIMIZED_SEQ_BUFFER_SIZE (<Sequence_Count>)

Note

A value of 0 will define the buffer size for fitting all optimize-able sequences

3.9.1.15 SpiEnableMultiSyncTransmit**Table 3-45. SpiEnableMultiSyncTransmit**

Description	Allow simultaneous calls to Spi_SyncTransmit() for different threads. Two concurrent calls to Spi_SyncTransmit() will be allowed only if the related sequences do not share HW units.
Class	Non AUTOSAR Requirement
Range	true, false
Default	false
Source File	Spi_Cfg.h
Source Representation	#define SPI_ENABLE_MULTI_SYNC_TRANSMIT (<STD_OFF, STD_ON>)

Note

Enabling this option will violate the following AUTOSAR requirements: SPI135, SPI114

3.9.1.16 SpiEnableHWUnitAsyncMode**Table 3-46. SpiEnableHWUnitAsyncMode**

Description	Enable Spi_SetHWUnitAsyncMode() function, which allows defining distinct operation mode (POLLING or INTERRUPT) for each HWUnit.
Class	Non AUTOSAR Requirement
Range	true, false
Default	false
Source File	Spi_Cfg.h
Source Representation	#define SPI_HWUNIT_ASYNC_MODE (<STD_OFF, STD_ON>)

Note

This feature is not required by Autosar, which defines asynchronous mode configuration at driver level only

3.9.1.17 SpiEnableDualClockMode

Table 3-47. SpiEnableDualClockMode

Description	Enable Spi_SetClockMode() function, which allows dual MCU clock configuration settings.
Class	Implementation specific parameter
Range	true, false
Default	false
Source File	Spi_Cfg.h
Source Representation	#define SPI_DUAL_CLOCK_MODE (<STD_OFF, STD_ON>)

Note

For each Job, the SPI driver holds two sets of timing parameters (namely the settings for the CTAR register) to achieve the specified Baudrate. One setting is the normal setting and the other one is the alternate setting. Each of these settings is derived from the relation of the Spi Baudrate and a corresponding Mcu Clock Reference Point. The Clock Reference Point for the normal setting is specified via the parameter "SpiClockRef" in Container "SpiGeneral". The Clock Reference Point for the alternate setting is specified via the parameter "SpiAlternateClockRef" in Container "SpiNonAUTOSAR". The appropriate setting must be activated at runtime by calling the function "Spi_SetClockMode" with the correct Clock Mode. Note: this function does not affect the actual Clock Generation as this is only set by the Mcu driver. It only tells the SPI driver which is the current active Clock Reference Point. So, in order to have correct SPI Bustiming, this function must be called always after Mcu_InitClock or Mcu_SetMode.

3.9.1.18 SpiAlternateClockRef

Table 3-48. SpiAlternateClockRef

Description	Reference to the alternate clock configuration, retrieved from the MCU plugin.
Class	Implementation specific parameter
Range	N/A
Default	N/A
Source File	Spi_Cfg.h
Source Representation	It is only used in code template.

3.9.1.19 SpiAllowBigSizeCollections

Table 3-49. SpiAllowBigSizeCollections

Description	Reference to the Big Size Collections configuration
Class	Implementation specific parameter
Range	N/A
Default	N/A
Source File	Spi_Cfg.h
Source Representation	#define SPI_ALLOW_BIGSIZE_COLLECTIONS (STD_OFF)

3.9.1.20 SpiJobStartNotificationEnable

Table 3-50. SpiStartJobNotificationEnable

Description	This feature is a non-Autosar feature to enable the job start notification.
Class	Implementation specific parameter
Range	True, False
Default	False
Source File	N/A
Source Representation	N/A

3.9.1.21 SpiTSBModeSupport

Table 3-51. SpiTSBModeSupport

Description	A feature to enable user to access the Micro Second Bus functionality
Class	Non-Autosar Parameter
Range	True, false
Default	false
Source File	Spi_Cfg.h
Source Representation	NA

3.9.1.22 TSBModeEnable

Table 3-52. TSBModeEnable

Description	This parameter defines the length of a dataframe. The TSBCNT field selects number of data bits to be shifted out during a transfer in TSB mode.
Class	Non-Autosar Parameter
Range	3-31
Default	16
Source File	Spi_PbCfg.c
Source Representation	#define SPI_TSB_MODE (STD_ON)

3.9.2 Post-Build parameters

Post-Build parameters, their possible values and their meaning are described in the following text. The Post-Build parameters are implemented as constant structures and arrays stored in flash memory of the MCU.

The configuration screen for SPI postbuild parameters in Tresos® Studio configuration tool from Tresos Tresos 2010a.sr4 20100415-release2010a-sr4 is given below:

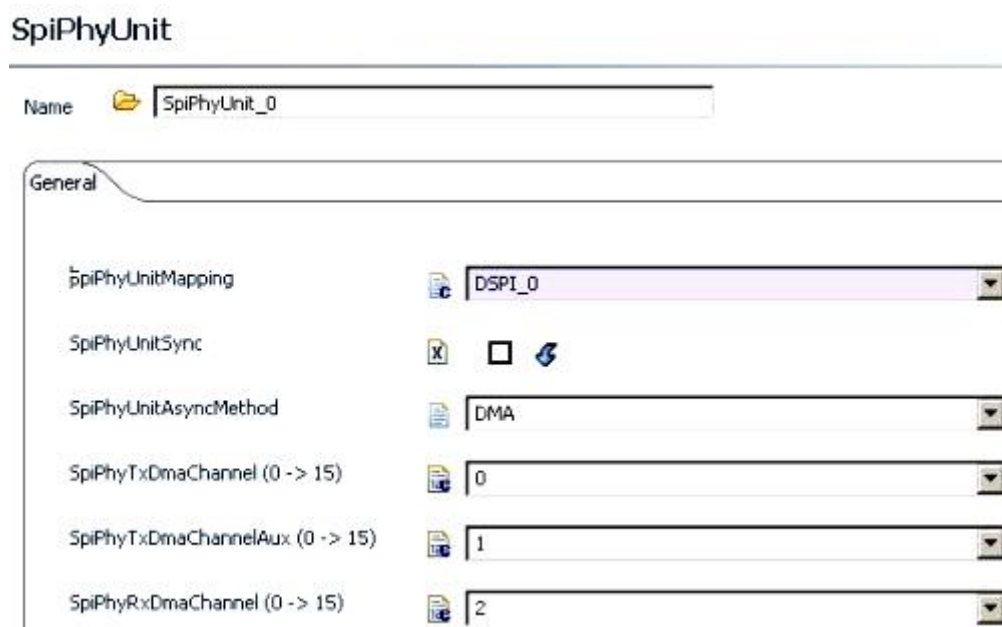


Figure 3-3. SpiPhyUnit Configuration

SpiChannel

Name*

General

SpiChannelId (0 -> 255)	<input type="text" value="0"/>	
SpiChannelType	<input type="text" value="IB"/>	
SpiDataWidth (4 -> 16)	<input type="text" value="8"/>	
SpiDefaultData (0 -> 65535)	<input type="text" value="1"/>	
SpiEbMaxLength (1 -> 65535)	<input type="text" value="1"/>	
SpiIbNbBuffers (1 -> 65535)	<input type="text" value="1"/>	
SpiTransferStart	<input type="text" value="MSB"/>	

Figure 3-4. SpiChannel Configuration

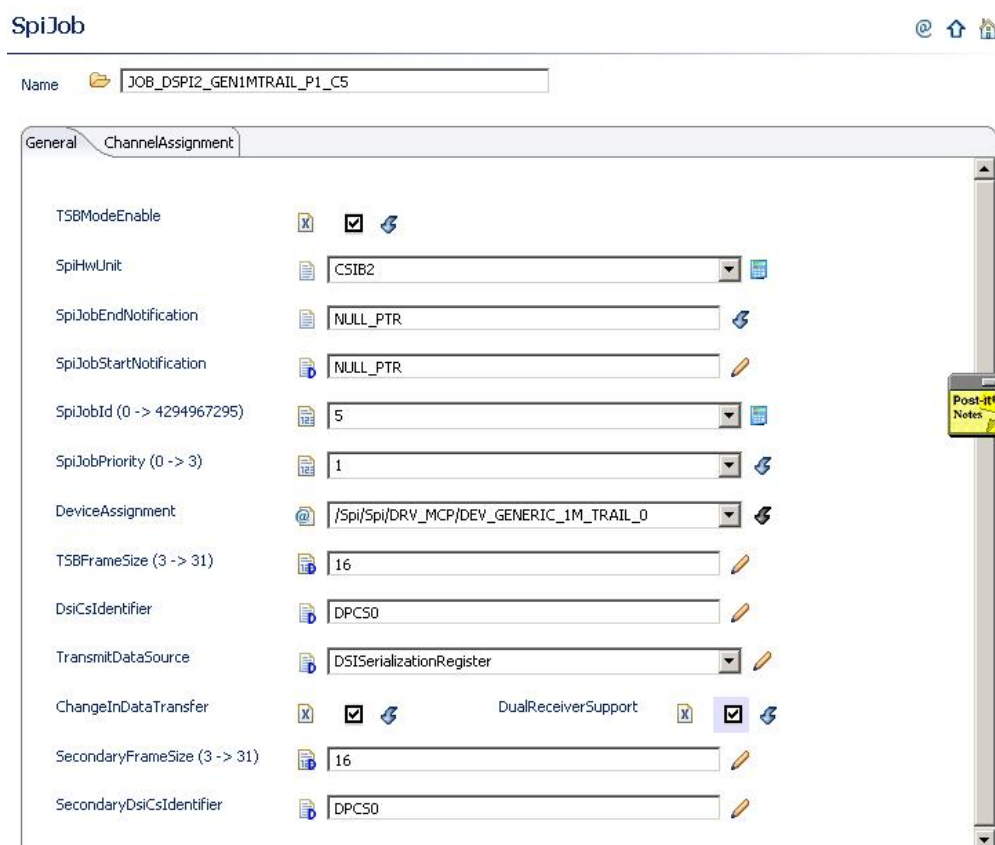


Figure 3-5. SpiJob Configuration

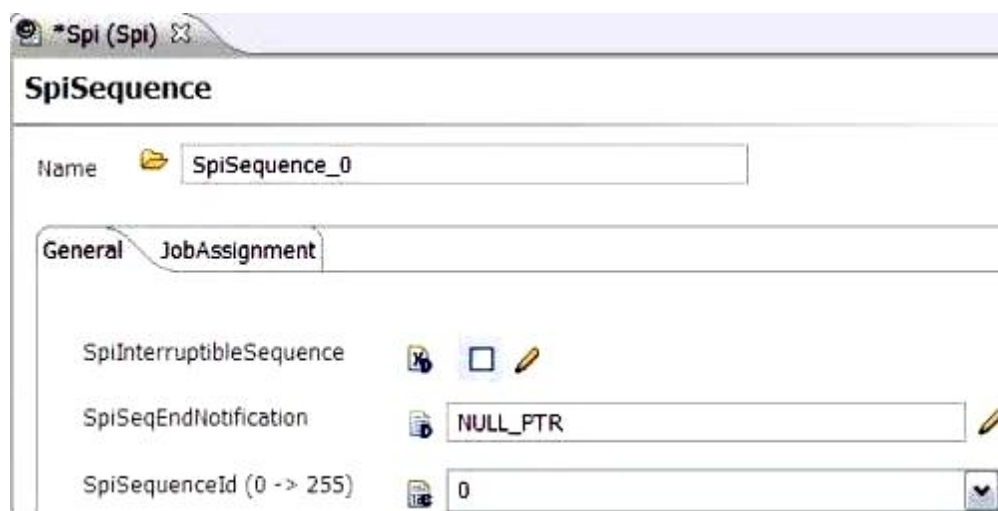


Figure 3-6. SpiSequence Configuration

***Spi (Spi)**

SpiExternalDevice

Name

General

SpiBaudrate (18 -> 107000000)	<input type="text" value="100000.0"/>
SpiCsIdentifier	<input type="text" value="PCS0"/>
SpiCsPolarity	<input type="text" value="HIGH"/>
SpiDataShiftEdge	<input type="text" value="LEADING"/>
SpiEnableCs	<input checked="" type="checkbox"/>
SpiShiftClockIdleLevel	<input type="text" value="HIGH"/>
SpiTimeClk2Cs (0 -> 100)	<input type="text" value="1.0"/>
SpiTimeCs2Clk (0 -> 10000)	<input type="text" value="1.0"/>
SpiTimeCs2Cs (0 -> 10000)	<input type="text" value="6.4"/>
SpiCsContinuous	<input type="text" value="true"/>

Figure 3-7. SpiExternalDevice Configuration

3.9.2.1 SpiPhyUnitMapping

Table 3-53. SpiPhyUnitMapping

Description	Logical SpiHWunit to physical DSPI_[0 1 2 3 4 5] assignment.
Class	Autosar parameter
Range	DSPI_0, DSPI_1
Default	DSPI_0
Source File	-

Table continues on the next page...

Table 3-53. SpiPhyUnitMapping (continued)

Source Representation	<p>The bold section of the following structure</p> <pre> static CONST(Spi_HWUnitConfig, SPI_CONST) HWUnitConfig_PB[SPI_MAX_HWUNIT] = { { DSPI_0_OFFSET, SPI_PHYUNIT_ASYNC, (uint16)TRUE, 14u, 27u, 15u }, </pre>
------------------------------	---

3.9.2.2 SpiPhyUnitSync

Table 3-54. SpiPhyUnitSync

Description	This parameter is used to specify if this HwUnit can do only synchronous transfers. If true then this hardware unit is dedicated for Synchronous transfers. If false then this hardware unit is dedicated for Asynchronous transfers.
Class	Autosar parameter
Range	True, false
Default	True
Source File	Spi_PbCfg.c
Source Representation	<p>The bold section of the following structure</p> <pre> static CONST(Spi_HWUnitConfig, SPI_CONST) HWUnitConfig_PB[SPI_MAX_HWUNIT] = { { DSPI_0_OFFSET, SPI_PHYUNIT_ASYNC, (uint16)TRUE, 14u, 27u, 15u }, </pre>

Note

Asynchronous Transfer (false) is applicable only if SpiLevelDelivered is configured for LEVEL1 or LEVEL2 and Synchronous Transfer (true) is applicable only if SpiLevelDelivered is configured for LEVEL0.

3.9.2.3 SpiPhyUnitAsyncMethod

Table 3-55. SpiPhyUnitAsyncMethod

Description	<p>This parameter is used only for Asynchronous Transmission.</p> <p>Select one of Asynchronous Transfer Method for Spi: DMA and PIO with FIFO.</p> <p>DMA - DMA is used as transfer method for transferring data from channel buffer to DSPI Transmit FIFO register and receives data from DSPI Receive FIFO register to channel buffer.</p> <p>PIO_FIFO - Programmed Input Output First in First Out – The Spi driver transfers a maximum of four data entries from channel buffer to DSPI Transmit FIFO registers and receives a maximum of four data entries from DSPI Receive FIFO registers to channel buffer.</p>
Class	Implementation specific parameter
Range	DMA, PIO_FIFO
Default	PIO_FIFO
Source File	Spi_Cfg.h
Source Representation	<p>The bold section of the following structure</p> <pre>static CONST(Spi_HWUnitConfig, SPI_CONST) HWUnitConfig_PB[SPI_MAX_HWUNIT] = { { DSPI_0_OFFSET, SPI_PHYUNIT_ASYNC, (uint16) TRUE, 14u, 27u, 15u }, </pre>

Note

For Asynchronous Transmission Notification function, SPI_JOB_FAILED will not be called from Spi Hardware since there are no conditions to detect a hardware failure and make the Job status as failed.

3.9.2.4 SpiPhyTxDmaChannel

Table 3-56. SpiPhyTxDmaChannel

Description	DSPI Master Transmit DMA Channel, used to prepare the DSPI transmission dataframes starting from the TX buffer content. A maximum of 0 to 15 channels could be configured.
Class	Implementation specific parameter
Range	-
Default	(DSPI_index x 3) + 0
Source File	Spi_Cfg.c, Spi_PBcfg.c, Spi_LTcfg.c

Table continues on the next page...

Table 3-56. SpiPhyTxDmaChannel (continued)

Source Representation	<p>The bold section of the following structure</p> <pre>static CONST(Spi_HWUnitConfig, SPI_CONST) HWUnitConfig_PB[SPI_MAX_HWUNIT] = { { DSPI_0_OFFSET, SPI_PHYUNIT_ASYNC, (uint16)TRUE, 14u, 27u, 15u }, </pre>
------------------------------	--

Note

SpiPhyTxDmaChannel is applicable only if DMA is selected for Asynchronous Transfer Method (SpiPhyUnitAsyncMethod).

3.9.2.5 SpiPhyTxDmaChannelAux**Table 3-57. SpiPhyTxDmaChannelAux**

Description	DSPI Auxiliary Transmit DMA Channel, used to place dataframes into the DSPI registers. A maximum of 0 to 15 channels could be configured.
Class	Implementation specific parameter
Range	-
Default	(DSPI_index x 3) + 1
Source File	Spi_Cfg.c, Spi_PBcfg.c, Spi_LTcfg.c
Source Representation	<p>The bold section of the following structure</p> <pre>static CONST(Spi_HWUnitConfig, SPI_CONST) HWUnitConfig_PB[SPI_MAX_HWUNIT] = { { DSPI_0_OFFSET, SPI_PHYUNIT_ASYNC, (uint16)TRUE, 14u, 27u, 15u }, </pre>

Note

SpiPhyTxDmaChannelAux is applicable only if DMA is selected for Asynchronous Transfer Method (SpiPhyUnitAsyncMethod).

3.9.2.6 SpiPhyRxDmaChannel

Table 3-58. SpiPhyRxDmaChannel

Description	DSPI Receive DMA Channel, used to read the deserialized dataframes into the RX buffers. A maximum of 0 to 15 channels could be configured.
Class	Implementation specific parameter
Range	-
Default	(DSPI_index x 3) + 2
Source File	Spi_Cfg.c, Spi_PBcfg.c, Spi_LTcfg.c
Source Representation	<p>The bold section of the following structure</p> <pre>static CONST(Spi_HWUnitConfig, SPI_CONST) HWUnitConfig_PB[SPI_MAX_HWUNIT] = { { DSPI_0_OFFSET, SPI_PHYUNIT_ASYNC, (uint16)TRUE, 14u, 27u, 15u }, </pre>

Note

SpiPhyRxDmaChannel is applicable only if DMA is selected for Asynchronous Transfer Method (SpiPhyUnitAsyncMethod).

3.9.2.7 ChannelAssignment

Table 3-59. ChannelAssignment

Description	A Job refers to several channels. This parameter forms a reference to several channels.
Class	Autosar Parameter
Range	NA
Default	-
Source File	Spi_PbCfg.c

Table continues on the next page...

Table 3-59. ChannelAssignment (continued)

Source Representation	<p>The bold portion of the following structure:</p> <pre> JOB_WREN_EEPROM1_ChannelAssignment_PB[1] = {CH_WREN_CMD_EEPROM1}; static CONST(Spi_JobConfig, SPI_CONST) SpiJobConfig_PB0[5] = { { /* JOB_WREN_EEPROM1 */ (Spi_ChannelType)1u, JOB_WREN_EEPROM1_ChannelAssignment_PB, /* List of Channels */ EndJob_WREN_EEPROM1, /* End Notification */ (sint8)0, /* Priority */ &SpiJobState[0], /* JobState instance */ CSIB0, DSPI_0_OFFSET, /* DSPI device HW unit offset */ /* External Device Settings */ EEPROM3_SPI2_CS0 } ... }, </pre>
------------------------------	---

3.9.2.8 SpiMaxChannel

Table 3-60. SpiMaxChannel

Description	This parameter contains the number of Channels configured. It will be gathered by tools during the configuration stage.
Class	Autosar Parameter
Range	-
Default	-
Source File	-
Source Representation	<pre> /* SpiDriverConfig_PB Configuration */ CONST(Spi_ConfigType, SPI_CONST) SpiDriverConfig_PB = { 1, 6, 5, 4, SpiChannelConfig_PB0, SpiJobConfig_PB0, SpiSequenceConfig_PB0, &SpiAttributesConfig_PB0, HWUnitConfig_PB }; </pre>

Note

SpiMaxChannel parameter is not used, instead max channel value is derived from number of channels configured.

SpiMaxChannel parameter is retained in configuration for consistency with AUTOSAR_EcucParamDef.xml rev 003.

3.9.2.9 SpiChannelId

Table 3-61. SpiChannelId

Description	Channel unique identifier in the configuration.
Class	Autosar Parameter
Range	0 – (SpiMaxChannel-1)
Default	0
Source File	-
Source Representation	#define <ChannelName> (Spi_ChannelType)<SpiChannelId>

3.9.2.10 SpiChannelType

Table 3-62. SpiChannelType

Description	Buffer usage with EB/IB channel
Class	Autosar Parameter
Range	IB , EB
Default	IB
Source File	-
Source Representation	<p>The bold section of following structure:</p> <pre> /* Channel Configuration */ static CONST(Spi_ChannelConfig, SPI_CONST) SpiChannelConfig_PB0[6] = { { /* CH_WREN_CMD_EEPROM1*/ IB, 6 1, &Buffer_CH_WREN_CMD_EEPROM1, &SpiChannelState[0] }, }; </pre>

3.9.2.11 SpiDataWidth

Table 3-63. SpiDataWidth

Description	This parameter is the width of a data unit to be transmitted.
Class	Autosar Parameter
Range	4...16
Default	8
Source File	Spi_PbCfg.c
Source Representation	<p>The bold section of the following structure:</p> <pre>static CONST(Spi_LLD_ChannelAttributesConfig, SPI_CONST) DspiChannelAttributesConfig_PB0[6] = { DSPI_CMD_MSB DSPI_CMD_WIDTH_8, DSPI_CMD_MSB DSPI_CMD_WIDTH_8, DSPI_CMD_MSB DSPI_CMD_WIDTH_8, DSPI_CMD_MSB DSPI_CMD_WIDTH_8 }</pre>

3.9.2.12 SpiDefaultData

Table 3-64. SpiDefaultData

Description	This parameter is the default value that is transmitted.
Class	Autosar Parameter
Range	-
Default	1
Source File	Spi_PbCfg.c
Source Representation	<p>The bold section of following structure:</p> <pre>/* Channel Configuration */ static CONST(Spi_ChannelConfig, SPI_CONST) SpiChannelConfig_PB0[6] = { { /* CH_WREN_CMD_EEPROM1*/ IB, 6, 1, &Buffer_CH_WREN_CMD_EEPROM1, &SpiChannelState[0] } }</pre>

3.9.2.13 SpiEbMaxlength

Table 3-65. SpiEbMaxlength

Description	This parameter contains the maximum size of data buffers in case of EB Channels only.
Class	Autosar Parameter
Range	-
Default	1
Source File	Spi_PbCfg.c
Source Representation	<p>The bold section of following structure:</p> <pre> /* Channel Configuration */ static CONST(Spi_ChannelConfig, SPI_CONST) SpiChannelConfig_PB0[6] = { { /* CH_WREN_CMD_EEPROM1*/ EB, 6, 1, &Buffer_CH_WREN_CMD_EEPROM1, &SpiChannelState[0] }, }; </pre>

3.9.2.14 SpilbNBuffers

Table 3-66. SpilbNBuffers

Description	This parameter contains the maximum number of data buffers in case of IB Channels and only. In case of SpiForceDataType ON and channel's Spi_DataWidth is between 9 and 16 bits, this parameter refers to the number of BYTES allocated to the buffers and MUST be even.
Class	Autosar Parameter
Range	1 to 65535
Default	65535
Source File	Spi_PbCfg.c
Source Representation	<p>The bold section of following structure:</p> <pre> /* Channel Configuration */ static CONST(Spi_ChannelConfig, SPI_CONST) SpiChannelConfig_PB0[6] = { { /* CH_WREN_CMD_EEPROM1*/ IB, 6, 1, &Buffer_CH_WREN_CMD_EEPROM1, &SpiChannelState[0] }, }; </pre>

3.9.2.15 SpiTransferStart

Table 3-67. SpiTransferStart

Description	This parameter defines the first starting bit for transmission.
Class	Autosar Parameter
Range	-
Default	-
Source File	Spi_PbCfg.c
Source Representation	<p>The bold section of the following structure:</p> <pre>static CONST(Spi_LLD_ChannelAttributesConfig, SPI_CONST) DspiChannelAttributesConfig_PB0[6] = { DSPI_CMD_MSB DSPI_CMD_WIDTH_8, DSPI_CMD_MSB DSPI_CMD_WIDTH_8, DSPI_CMD_MSB DSPI_CMD_WIDTH_8, DSPI_CMD_MSB DSPI_CMD_WIDTH_8, }</pre>

3.9.2.16 SpiBaudRate

Table 3-68. SpiBaudRate

Description	This parameter is the communication baudrate – This parameter allows using a range of values, from the point of view of configuration tools. SpiBaudrate is dependant on Processor Frequency. The minimum baud rate at 4Mhz Processor frequency is 18 bits / second (18 bits/s) and the maximum baud rate at 64 MHz Processor frequency is 10,700,000 bits/second (10.7Mb/s)
Class	Autosar Parameter
Range	18 bits/second – 10,700,000 bits/second
Default	100000
Source File	Spi_PbCfg.c

Table continues on the next page...

Table 3-68. SpiBaudRate (continued)

Source Representation	<p>The bold section of the following structure</p> <pre> static CONST(Spi_LLD_DeviceAttributesConfig, SPI_CONST) DspiDeviceAttributesConfig_PB0[1] = { { (uint32)(DSPI_CTAR_CPOL_HIGH DSPI_CTAR_CPHA_LEADING DSPI_CTAR_PCSCLK_3 DSPI_CTAR_SCCLK_64 DSPI_CTAR_PASC_3 DSPI_CTAR_SASC_128 DSPI_CTAR_PDT_3 DSPI_CTAR_SDT_64 DSPI_CTAR_PBR_5 DSPI_CTAR_SBR_16 DSPI_CTAR_DBR_0), (uint32)(DSPI_CMD_PCS0 /* Chip Select Pin */ DSPI_CMD_CONTINUOUS_TRUE), /* Continuous chip select */ (uint32)DSPI_PCS0_IDLEHIGH /* Chip select polarity */ }, }; </pre>
------------------------------	---

3.9.2.17 SpiCs

Table 3-69. SpiCs

Description	This parameter defines the Chip select pin
Class	Autosar Parameter
Range	PCS0 to PCS1
Default	PCS0
Source File	Spi_PbCfg.c
Source Representation	<p>The bold section of the following structure</p> <pre> static CONST(Spi_LLD_DeviceAttributesConfig, SPI_CONST) DspiDeviceAttributesConfig_PB0[1] = { { (uint32)(DSPI_CTAR_CPOL_HIGH DSPI_CTAR_CPHA_LEADING DSPI_CTAR_PCSCLK_3 DSPI_CTAR_SCCLK_64 DSPI_CTAR_PASC_3 DSPI_CTAR_SASC_128 DSPI_CTAR_PDT_3 DSPI_CTAR_SDT_64 DSPI_CTAR_PBR_5 DSPI_CTAR_SBR_16 DSPI_CTAR_DBR_0), (uint32)(DSPI_CMD_PCS0 /* Chip Select Pin */ DSPI_CMD_CONTINUOUS_TRUE), /* Continuous chip select */ (uint32)DSPI_PCS0_IDLEHIGH /* Chip select polarity */ }, }; </pre>

3.9.2.18 SpiCsPolarity

Table 3-70. SpiCsPolarity

Description	This parameter defines the active polarity of Chip Select.
Class	Autosar Parameter
Range	High / Low
Default	High
Source File	Spi_PbCfg.c
Source Representation	<p>The bold section of the following structure:</p> <pre>static CONST(Spi_LLD_DeviceAttributesConfig,SPI_CONST) DspiDeviceAttributesConfig_PB0[1] = { { (uint32)(DSPI_CTAR_CPOL_HIGH DSPI_CTAR_CPHA_LEADING DSPI_CTAR_PCCLK_3 DSPI_CTAR_SCCLK_64 DSPI_CTAR_PASC_3 DSPI_CTAR_SASC_128 DSPI_CTAR_PDT_3 DSPI_CTAR_SDT_64 DSPI_CTAR_PBR_5 DSPI_CTAR_SBR_16 DSPI_CTAR_DBR_0), (uint32)(DSPI_CMD_PCS0 /* Chip Select Pin */ DSPI_CMD_CONTINUOUS_TRUE), /* Continous chip select */ (uint32)DSPI_PCS0_IDLEHIGH/* Chip select polarity */ }, };</pre>

3.9.2.19 SpiDataShiftEdge

Table 3-71. SpiDataShiftEdge

Description	This parameter defines the SPI data shift edge.
Class	Autosar Parameter
Range	Leading / Trailing
Default	Leading
Source File	Spi_PbCfg.c

Table continues on the next page...

Table 3-71. SpiDataShiftEdge (continued)

Source Representation	<p>The bold section of the following structure</p> <pre> static CONST(Spi_LLD_DeviceAttributesConfig,SPI_CONST) DspiDeviceAttributesConfig_PB0[1] = { { (uint32) (DSPI_CTAR_CPOL_HIGH DSPI_CTAR_CPHA_LEADING DSPI_CTAR_PCSClk_3 DSPI_CTAR_SCSClk_64 DSPI_CTAR_PASC_3 DSPI_CTAR_SASC_128 DSPI_CTAR_PDT_3 DSPI_CTAR_SDT_64 DSPI_CTAR_PBR_5 DSPI_CTAR_SBR_16 DSPI_CTAR_DBR_0), (uint32) (DSPI_CMD_PCS0 /* Chip Select Pin */ DSPI_CMD_CONTINUOUS_TRUE), /* Continous chip select */ (uint32) DSPI_PCS0_IDLEHIGH /* Chip select polarity */ }, }; </pre>
------------------------------	---

3.9.2.20 SpiEnableCs

Table 3-72. SpiEnableCs

Description	This parameter enables or not the Chip Select handling functions. If True then Chip Select signal is generated and if False then the Chip Select signal would not be generated.
Class	Autosar Parameter
Range	True / False
Default	False
Source File	Spi_PbCfg.c
Source Representation	<p>The bold section of the following structure:</p> <pre> static CONST(Spi_LLD_DeviceAttributesConfig,SPI_CONST) DspiDeviceAttributesConfig_PB0[1] = { { (uint32) (DSPI_CTAR_CPOL_HIGH DSPI_CTAR_CPHA_LEADING DSPI_CTAR_PCSClk_3 DSPI_CTAR_SCSClk_64 DSPI_CTAR_PASC_3 DSPI_CTAR_SASC_128 DSPI_CTAR_PDT_3 DSPI_CTAR_SDT_64 DSPI_CTAR_PBR_5 DSPI_CTAR_SBR_16 DSPI_CTAR_DBR_0), (uint32) (DSPI_CMD_PCS0 /* Chip Select Pin */ DSPI_CMD_CONTINUOUS_TRUE), /* Continous chip select */ (uint32) DSPI_PCS0_IDLEHIGH /* Chip select polarity */ }, }; </pre>

Note

When SpiEnableCs is true, the corresponding chip select and SpiCsContinuous parameter values would be generated in the configuration and CS signal will be handled by SPI driver. If SpiEnableCs is false, the value generated by the configuration

tool would be 0x00000000. When SpiEnableCs is disabled, the driver would not generate the CS at all, so in that case customer should take care of chip select handling in their software, and it will work only in case a sequence is configured to handle jobs on a unique device.

3.9.2.21 SpiShiftClockIdleLevel

Table 3-73. SpiShiftClockIdleLevel

Description	This parameter defines the SPI shift clock idle level.
Class	Autosar Parameter
Range	High/ Low
Default	High
Source File	Spi_PbCfg.c
Source Representation	<p>The bold section of the following structure:</p> <pre>static CONST(Spi_LLD_DeviceAttributesConfig, SPI_CONST) DspiDeviceAttributesConfig_PB0[1] = { { (uint32)(DSPI_CTAR_CPOL_HIGH DSPI_CTAR_CPHA_LEADING DSPI_CTAR_PCCLK_3 DSPI_CTAR_SCCLK_64 DSPI_CTAR_PASC_3 DSPI_CTAR_SASC_128 DSPI_CTAR_PDT_3 DSPI_CTAR_SDT_64 DSPI_CTAR_PBR_5 DSPI_CTAR_SBR_16 DSPI_CTAR_DBR_0), (uint32)(DSPI_CMD_PCS0 /* Chip Select Pin */ DSPI_CMD_CONTINUOUS_TRUE), /* Continuous chip select */ (uint32)DSPI_PCS0_IDLEHIGH/* Chip select polarity */ }, };</pre>

3.9.2.22 SpiTimeClk2Cs

Table 3-74. SpiTimeClk2Cs

Description	This parameter is the minimum time between clock and Chip Select.
Class	Autosar Parameter
Range	0 to 100 (µs)
Default	1.0
Source File	Spi_PbCfg.c

Table continues on the next page...

Table 3-74. SpiTimeClk2Cs (continued)

Source Representation	<p>The bold section of the following structure:</p> <pre> static CONST(Spi_LLD_DeviceAttributesConfig, SPI_CONST) DspiDeviceAttributesConfig_PB0[1] = { { (uint32) (DSPI_CTAR_CPOL_HIGH DSPI_CTAR_CPHA_LEADING DSPI_CTAR_PCCLK_3 DSPI_CTAR_SCCLK_64 DSPI_CTAR_PASC_3 DSPI_CTAR_SASC_128 DSPI_CTAR_PDT_3 DSPI_CTAR_SDT_64 DSPI_CTAR_PBR_5 DSPI_CTAR_SBR_16 DSPI_CTAR_DBR_0), (uint32) (DSPI_CMD_PCS0 /* Chip Select Pin */ DSPI_CMD_CONTINUOUS_TRUE), /* Continuous chip select */ (uint32) DSPI_PCS0_IDLEHIGH /* Chip select polarity */ }, }; </pre>
------------------------------	---

3.9.2.23 SpiTimeCs2Clk

Table 3-75. SpiTimeCs2Clk

Description	This parameter is the time between Chip Select and next assertion of the clock
Class	Autosar Parameter
Range	0 to 10000 (μs)
Default	1.0
Source File	Spi_PbCfg.c
Source Representation	<p>The bold section of the following structure:</p> <pre> static CONST(Spi_LLD_DeviceAttributesConfig, SPI_CONST) DspiDeviceAttributesConfig_PB0[1] = { { (uint32) (DSPI_CTAR_CPOL_HIGH DSPI_CTAR_CPHA_LEADING DSPI_CTAR_PCCLK_3 DSPI_CTAR_SCCLK_64 DSPI_CTAR_PASC_3 DSPI_CTAR_SASC_128 DSPI_CTAR_PDT_3 DSPI_CTAR_SDT_64 DSPI_CTAR_PBR_5 DSPI_CTAR_SBR_16 DSPI_CTAR_DBR_0), (uint32) (DSPI_CMD_PCS0 /* Chip Select Pin */ DSPI_CMD_CONTINUOUS_TRUE), /* Continuous chip select */ (uint32) DSPI_PCS0_IDLEHIGH /* Chip select polarity */ }, }; </pre>

3.9.2.24 SpiTimeCs2Cs

Table 3-76. SpiTimeCs2Cs

Description	This parameter defines the time between the Chip selects assertion in microseconds.
Class	Implementation specific parameter
Range	0 to 10000(μs)
Default	6.4
Source File	Spi_PbCfg.c
Source Representation	<p>The bold section of the following structure:</p> <pre>static CONST(Spi_LLD_DeviceAttributesConfig, SPI_CONST) DspiDeviceAttributesConfig_PB0[1] = { { (uint32)(DSPI_CTAR_CPOL_HIGH DSPI_CTAR_CPHA_LEADING DSPI_CTAR_PCSCLK_3 DSPI_CTAR_SCCLK_64 DSPI_CTAR_PASC_3 DSPI_CTAR_SASC_128 DSPI_CTAR_PDT_3 DSPI_CTAR_SDT_64 DSPI_CTAR_PBR_5 DSPI_CTAR_SBR_16 DSPI_CTAR_DBR_0), (uint32)(DSPI_CMD_PCS0 /* Chip Select Pin */ DSPI_CMD_CONTINUOUS_TRUE), /* Continous chip select */ (uint32)DSPI_PCS0_IDLEHIGH /* Chip select polarity */ }, };</pre>

3.9.2.25 SpiCsContinuous

Table 3-77. SpiCsContinuous

Description	This parameter determines to keep chip selected between frame transfers.
Class	Implementation specific parameter
Range	True / false
Default	True
Source File	Spi_PbCfg.c

Table continues on the next page...

**Table 3-77. SpiCsContinuous
(continued)**

Source Representation	<p>The bold section of the following structure:</p> <pre> static CONST(Spi_LLD_DeviceAttributesConfig, SPI_CONST) DspiDeviceAttributesConfig_PB0[1] = { { (uint32)(DSPI_CTAR_CPOL_HIGH DSPI_CTAR_CPHA_LEADING DSPI_CTAR_PCSClk_3 DSPI_CTAR_SCSClk_64 DSPI_CTAR_PASC_3 DSPI_CTAR_SASC_128 DSPI_CTAR_PDT_3 DSPI_CTAR_SDT_64 DSPI_CTAR_PBR_5 DSPI_CTAR_SBR_16 DSPI_CTAR_DBR_0), (uint32)(DSPI_CMD_PCS0 /* Chip Select Pin */ DSPI_CMD_CONTINUOUS_TRUE), /* Continuous chip select */ (uint32)DSPI_PCS0_IDLEHIGH /* Chip select polarity */ }, }; </pre>
------------------------------	---

3.9.2.26 SpiMaxJob

Table 3-78. SpiMaxJob

Description	This parameter contains the number of Jobs configured. It will be gathered by tools during the configuration stage.
Class	Autosar Parameter
Range	-
Default	-
Source File	-
Source Representation	<p>The bold section of following structure:</p> <pre> /* SpiDriverConfig_PB Configuration */ CONST(Spi_ConfigType, SPI_CONST) SpiDriverConfig_PB = { 1, 6, 5, 4, SpiChannelConfig_PB0, SpiJobConfig_PB0, SpiSequenceConfig_PB0, &SpiAttributesConfig_PB0, HWUnitConfig_PB }; </pre>

Note

SpiMaxJob parameter is not used, instead max jobs value is derived from number of jobs configured. SpiMaxJob parameter is retained in configuration for consistency with AUTOSAR_EcucParamDef.xml rev 003.

3.9.2.27 SpiHwUnit

Table 3-79. SpiHwUnit

Description	This parameter is the symbolic name to identify the HW SPI Hardware microcontroller peripheral allocated to this Job.
Class	Autosar Parameter
Range	CSIB0, CSIB1, CSIB2, CSIB3, CSIB4, CSIB5
Default	-
Source File	Spi_PbCfg.c
Source Representation	<p>The bold section of following structure:</p> <pre> static CONST(Spi_JobConfig, SPI_CONST) SpiJobConfig_PB0[5] = { { /* JOB_WREN_EEPROM1 */ (Spi_ChannelType)1u, JOB_WREN_EEPROM1_ChannelAssignment_PB, /* List of Channels */ EndJob_WREN_EEPROM1, /* End Notification */ (sint8)0, /* Priority */ &SpiJobState[0], /* JobState instance */ CSIB0, DSPI_0_OFFSET, /* DSPI device HW unit offset */ /* External Device Settings */ EEPROM3_SPI2_CS0 { ... } }, }; </pre>

3.9.2.28 SpiJobId

Table 3-80. SpiJobId

Description	Job unique identifier in the configuration.
Class	Autosar Parameter
Range	0 – (SpiMaxJob-1)
Default	0
Source File	-
Source Representation	#define <JobName> ((Spi_JobType)<SpiJobId>)

3.9.2.29 SpiJobPriority

Table 3-81. SpiJobPriority

Description	This parameter defines the Job priority.
Class	Autosar Parameter
Range	0 – 3
Default	0
Source File	-
Source Representation	<p>The bold section of the following structure</p> <pre>static CONST(Spi_JobConfig, SPI_CONST) SpiJobConfig_PB0[5] = { { /* JOB_WREN_EEPROM1 */ (Spi_ChannelType)1u, JOB_WREN_EEPROM1_ChannelAssignment_PB, /* List of Channels */ EndJob_WREN_EEPROM1, /* End Notification */ (sint8)0, /* Priority */ &SpiJobState[0], /* JobState instance */ CSIB0, DSPI_0_OFFSET, /* DSPI device HW unit offset */ /* External Device Settings */ EEPROM3_SPI2_CS0 { ... } }, };</pre>

3.9.2.30 DeviceAssignment

Table 3-82. DeviceAssignment

Description	This parameter forms a reference to the external device used by the Job.
Class	Autosar Parameter
Range	NA
Default	-
Source File	Spi_PbCfg.c

Table continues on the next page...

Table 3-82. DeviceAssignment (continued)

Source Representation	<p>The bold section of the following structure</p> <pre> static CONST(Spi_JobConfig, SPI_CONST) SpiJobConfig_PB0[5] = { { /* JOB_WREN_EEPROM1 */ (Spi_ChannelType)1u, JOB_WREN_EEPROM1_ChannelAssignment_PB, /* List of Channels */ EndJob_WREN_EEPROM1, /* End Notification */ (uint8)0, /* Priority */ &SpiJobState[0], /* JobState instance */ CSIB0, DSPI_0_OFFSET, /* DSPI device HW unit offset */ /* External Device Settings */ EEPROM3_SPI2_CS0 { ... } }, }; </pre>
------------------------------	--

3.9.2.31 SpiMaxSequence

Table 3-83. SpiMaxSequence

Description	This parameter contains the number of Sequences configured. It will be gathered by tools during the configuration stage.
Class	Autosar Parameter
Range	-
Default	-
Source File	-
Source Representation	<p>The bold section of following structure:</p> <pre> /* SpiDriverConfig_PB Configuration */ CONST(Spi_ConfigType, SPI_CONST) SpiDriverConfig_PB = { 1, 6, 5, 4, SpiChannelConfig_PB0, SpiJobConfig_PB0, SpiSequenceConfig_PB0, &SpiAttributesConfig_PB0, HWUnitConfig_PB }; </pre>

Note

SpiMaxSequence parameter is not used, instead max Sequences value is derived from number of Sequences configured.

SpiMaxSequence parameter is retained in configuration for consistency with AUTOSAR_EcucParamDef.xml rev 003.

3.9.2.32 SpiInterruptibleSequence

Table 3-84. SpiInterruptibleSequence

Description	This parameter determines whether to allow or not this Sequence to be suspended by another one.
Class	Autosar Parameter
Range	True/False
Default	-
Source File	Spi_PbCfg.c
Source Representation	<p>The bold section of the following structure:</p> <pre>static CONST(Spi_SequenceConfig, SPI_CONST) SpiSequenceConfig_PB0[4] = { { (uint16)1u, SEQ_WREN_EEPROM1_JobAssignment_PB,/* List of Jobs */ Spi_Notification_Seq1,/* End Notification */ FALSE/* Interruptible */ }, };</pre>

3.9.2.33 SpiSeqEndNotification

Table 3-85. SpiSeqEndNotification

Description	This parameter is a reference to a notification function.
Class	Autosar Parameter
Range	-
Default	-
Source File	Spi_PbCfg.c
Source Representation	<p>The bold section of the following structure:</p> <pre>static CONST(Spi_SequenceConfig, SPI_CONST) SpiSequenceConfig_PB0[4] = { { (uint16)1u, SEQ_WREN_EEPROM1_JobAssignment_PB,/* List of Jobs */ Spi_Notification_Seq1,/* End Notification */ FALSE /* Interruptible */ }, };</pre>

3.9.2.34 SpiSequenceld

Table 3-86. SpiSequenceld

Description	Sequence unique identifier in the configuration.
Class	Autosar Parameter
Range	0 – (SpiMaxSequence-1)
Default	0
Source File	-
Source Representation	#define <SequenceName> ((Spi_SequenceType)<SpiSequenceId>)

3.9.2.35 JobAssignment

Table 3-87. JobAssignment

Description	A sequence references several jobs, which are executed during a communication sequence. This parameter forms a reference to several Jobs.
Class	Autosar Parameter
Range	NA
Default	-
Source File	Spi_PbCfg.c
Source Representation	<p>The bold section of the following structure</p> <pre>static CONST(Spi_JobType, SPI_CONST) SEQ_WREN_EEPROM1_JobAssignment_PB[1] = {JOB_WREN_EEPROM1}; static CONST(Spi_SequenceConfig, SPI_CONST) SpiSequenceConfig_PB0[4] = { { /* SEQ_WREN_EEPROM1 */ (uint16)1u, SEQ_WREN_EEPROM1_JobAssignment_PB, /* List of Jobs */ Spi_Notification_Seq1, /* End Notification */ FALSE/* Interruptible */ }, };</pre>

3.9.2.36 SpiJobEndNotification

Table 3-88. SpiJobEndNotification

Description	This parameter is a reference to a notification function. This function is editable only if SpiLevelDelivered is configured for either LEVEL1 or LEVEL2.
Class	Autosar Parameter

Table continues on the next page...

**Table 3-88. SpiJobEndNotification
(continued)**

Range	-
Default	-
Source File	Spi_PbCfg.c
Source Representation	<p>The bold section of the following structure:</p> <pre> static CONST(Spi_JobConfig, SPI_CONST) SpiJobConfig_PB0[5] = { { /* JOB_WREN_EEPROM1 */ (Spi_ChannelType)1u, JOB_WREN_EEPROM1_ChannelAssignment_PB, /* List of Channels */ EndJob_WREN_EEPROM1, /* End Notification */ (sint8)0, /* Priority */ &SpiJobState[0], /* JobState instance */ CSIB0, DSPI_0_OFFSET, /* DSPI device HW unit offset */ /* External Device Settings */ EEPROM3_SPI2_CS0 } ... }; </pre>

3.9.2.37 SpiJobstartNotification

Table 3-89. SpijobstartNotification

Description	This parameter is a reference to a notification function.
Class	Non-Autosar Parameter
Range	NA
Default	-
Source File	Spi_PBCfg.c

Table continues on the next page...

Table 3-89. SpijobstartNotification (continued)

Source Representation	<p>The bold section of the following structure</p> <pre> /* SpiJobConfig_PB0 Job Configuration of DRV_MCP*/ static CONST(Spi_JobConfig, SPI_CONST) SpiJobConfig_PB0[58] = { { /* JOB_DSPI1_EXP_P1_C67_N */ (Spi_ChannelType)1u, JOB_DSPI1_EXP_P1_C67_N ChannelAssignment_PB, /* List of Channels */ EndJob_DSPI1_EXP_P1_C67, /* End Notification */ StartJob_DSPI1_EXP_C67, /* job Start Notification */ (sint8)1, /* Priority */ &SpiJobState[0], /* JobState instance */ CSIB1, /* HWUnit index */ DSPI_1_OFFSET, /* DSPI device HW unit offset */ /* External Device Settings */ DEV_EXP_100K_LEAD_0, /* External Device */ { (uint32)(DSPI_CTAR_CPOL_LOW /* Clock Polarity (Idle State) */ DSPI_CTAR_CPHA_LEADING /* Clock Phase */ DSPI_CTAR_PCCLK_3 DSPI_CTAR_SCCLK_512 /* TimeCs2Clk: Should=100.0, Is=96, Error=-4.0% */ DSPI_CTAR_PASC_3 DSPI_CTAR_SASC_256 /* TimeClk2Cs: Should=50.0, Is=48, Error=-4.0% */ DSPI_CTAR_PDT_3 DSPI_CTAR_SDT_512 /* TimeCs2Cs: Should=100.0, Is=96, Error=-4.0% */ DSPI_CTAR_PBR_5 DSPI_CTAR_SBR_32 DSPI_CTAR_DBR_0), /* Baudrate: Should=100000.0, Is=100000, Error=0.0% */ 0x00000000u, (uint32)0u /* Chip select polarity */ } }, }; </pre>
------------------------------	---

3.9.2.38 DsiCsIdentifier

Table 3-90. DsiCsIdentifier

Description	This parameter is the symbolic name to identify the Chip Select (CS) allocated to this Job in DSI mode.
Class	Non-Autosar Parameter
Range	DPCS0-DPCS7
Default	DPCS0
Source File	Spi_PbCfg.c

Table continues on the next page...

Table 3-90. DsiCsIdentifier (continued)

Source Representation	<p>The bold section of the following structure</p> <pre> CONST(Spi_TSBCfg, SPI_CONST) SpiJobTSBCfg[1] = { { (Spi_JobType)(0), /*Normal Job Id*/ /*DSICR attributes*/ (uint32)(DSPI_DSICR_CS_DPCS0 DSPI_DSICR_TXSS_CLEAR DSPI_DSICR_CID_CLEAR), /*DSICR1 attributes*/ (uint32)(DSPI_DSICR1_CS_DPCS0 DSPI_DSICR1_TSBCNT_31), /*CTAR2 attributes*/ /* External Device Settings */ (uint32)((DSPI_CTAR_CPOL_HIGH /* Clock Polarity (Idle State) */ DSPI_CTAR_CPHA_LEADING /* Clock Phase */ DSPI_CTAR_PCCLK_5 DSPI_CTAR_SCCLK_16 /* TimeCs2Clk: Should=1.0, Is=1, Error=0.0% */ DSPI_CTAR_PASC_5 DSPI_CTAR_SASC_16 /* TimeClk2Cs: Should=1.0, Is=1, Error=0.0% */ DSPI_CTAR_PDT_1 DSPI_CTAR_SDT_512 /* TimeCs2Cs: Should=6.4, Is=6, Error=0.0% */ DSPI_CTAR_PBR_3 DSPI_CTAR_SBR_256 DSPI_CTAR_DBR_0) /* Baudrate: Should=100000.0, Is=104166, Error=4.17% */ (DSPI_CMD_WIDTH_16)) } }; </pre>
------------------------------	---

3.9.2.39 SecondaryDsiCsIdentifier

Table 3-91. SecondaryDsiCsIdentifier

Description	This parameter is the symbolic name to identify the Chip Select (CS) allocated to this Job in DSI mode in dual receiver mode.
Class	Non-Autosar Parameter
Range	DPCS0-DPCS7
Default	DPCS0
Source File	Spi_PbCfg.c

Table continues on the next page...

Table 3-91. SecondaryDsiCsIdentifier (continued)

Source Representation	<p>The bold section of the following structure:</p> <pre> CONST(Spi_TSBCConfig, SPI_CONST) SpiJobTSBCConfig[1] = { { (Spi_JobType)(0), /*Normal Job Id*/ /*DSICR attributes*/ (uint32)(DSPI_DSICR_CS_DPCS0 DSPI_DSICR_TXSS_CLEAR DSPI_DSICR_CID_CLEAR), /*DSICR1 attributes*/ (uint32)(DSPI_DSICR1_CS_DPCS0 DSPI_DSICR1_TSBCNT_31), /*CTAR2 attributes*/ /* External Device Settings */ (uint32)((DSPI_CTAR_CPOL_HIGH /* Clock Polarity (Idle State) */ DSPI_CTAR_CPHA_LEADING /* Clock Phase */ DSPI_CTAR_PCCLK_5 DSPI_CTAR_SCCLK_16 /* TimeCs2Clk: Should=1.0, Is=1, Error=0.0% */ DSPI_CTAR_PASC_5 DSPI_CTAR_SASC_16 /* TimeClk2Cs: Should=1.0, Is=1, Error=0.0% */ DSPI_CTAR_PDT_1 DSPI_CTAR_SDT_512 /* TimeCs2Cs: Should=6.4, Is=6, Error=0.0% */ DSPI_CTAR_PBR_3 DSPI_CTAR_SBR_256 DSPI_CTAR_DBR_0) /* Baudrate: Should=100000.0, Is=104166, Error=4.17% */ (DSPI_CMD_WIDTH_16)) } }; </pre>
------------------------------	--

3.9.2.40 SecondaryFrameSize

Table 3-92. SecondaryFrameSize

Description	The FMSZ field vaule(+1) equal the data frame bit number, where the control of PCS switches from the DSPI_DSICR to DSPI_CR1 register
Class	Non-Autosar Parameter
Range	3-31
Default	16
Source File	Spi_PbCfg.c

Table continues on the next page...

Table 3-92. SecondaryFrameSize (continued)

Source Representation	<p>The bold section of the following structure:</p> <pre> CONST(Spi_TSBCConfig, SPI_CONST) SpiJobTSBConfig[1] = { { (Spi_JobType)(0), /*Normal Job Id*/ /*DSICR attributes*/ (uint32)(DSPI_DSICR_CS_DPCS0 DSPI_DSICR_TXSS_CLEAR DSPI_DSICR_CID_CLEAR), /*DSICR1 attributes*/ (uint32)(DSPI_DSICR1_CS_DPCS0 DSPI_DSICR1_TSBcnt_31), /*CTAR2 attributes*/ /* External Device Settings */ (uint32)((DSPI_CTAR_CPOL_HIGH /* Clock Polarity (Idle State) */ DSPI_CTAR_CPHA_LEADING /* Clock Phase */ DSPI_CTAR_PCCLK_5 DSPI_CTAR_SCCLK_16 /* TimeCs2Clk: Should=1.0, Is=1, Error=0.0% */ DSPI_CTAR_PASC_5 DSPI_CTAR_SASC_16 /* TimeClk2Cs: Should=1.0, Is=1, Error=0.0% */ DSPI_CTAR_PDT_1 DSPI_CTAR_SDT_512 /* TimeCs2Cs: Should=6.4, Is=6, Error=0.0% */ DSPI_CTAR_PBR_3 DSPI_CTAR_SBR_256 DSPI_CTAR_DBR_0) /* Baudrate: Should=100000.0, Is=104166, Error=4.17% */ (DSPI_CMD_WIDTH_16)) } }; </pre>
------------------------------	---

3.9.2.41 TransmitDataSource

Table 3-93. TransmitDataSource

Description	This parameter selects the source register of data to be serialized. The source can be either data from host software (serialization data register) or parallel input pins states latched into the serialization data register..
Class	Non-Autosar Parameter
Range	NA
Default	DSISerializationRegister
Source File	Spi_PbCfg.c

Table continues on the next page...

Table 3-93. TransmitDataSource (continued)

Source Representation	<p>The bold section of the following structure:</p> <pre> CONST(Spi_TSBConfig, SPI_CONST) SpiJobTSBConfig[1] = { { (Spi_JobType)(0), /*Normal Job Id*/ /*DSICR attributes*/ (uint32)(DSPI_DSICR_CS_DPCS0 DSPI_DSICR_TXSS_CLEAR DSPI_DSICR_CID_CLEAR), /*DSICR1 attributes*/ (uint32)(DSPI_DSICR1_CS_DPCS0 DSPI_DSICR1_TSB_CNT_31), /*CTAR2 attributes*/ /* External Device Settings */ (uint32)((DSPI_CTAR_CPOL_HIGH /* Clock Polarity (Idle State) */ DSPI_CTAR_CPHA_LEADING /* Clock Phase */ DSPI_CTAR_PCCLK_5 DSPI_CTAR_SCCLK_16 /* TimeCs2Clk: Should=1.0, Is=1, Error=0.0% */ DSPI_CTAR_PASC_5 DSPI_CTAR_SASC_16 /* TimeClk2Cs: Should=1.0, Is=1, Error=0.0% */ DSPI_CTAR_PDT_1 DSPI_CTAR_SDT_512 /* TimeCs2Cs: Should=6.4, Is=6, Error=0.0% */ DSPI_CTAR_PBR_3 DSPI_CTAR_SBR_256 DSPI_CTAR_DBR_0) /* Baudrate: Should=100000.0, Is=104166, Error=4.17% */ (DSPI_CMD_WIDTH_16)) } }; </pre>
------------------------------	--

3.9.2.42 TSBFrameSize

Table 3-94. TSBFrameSize

Description	This parameter defines the length of a dataframe. The TSB_CNT field selects number of data bits to be shifted out during a transfer in TSB mode. Note:The hardware supports data width from 3 to 31 bit.The unit is in bits.
Class	Non-Autosar Parameter
Range	true,false
Default	false
Source File	Spi_PbCfg.c

Table continues on the next page...

Table 3-94. TSBFrameSize (continued)

Source Representation	<p>The bold section of the following structure</p> <pre> CONST(Spi_TSBConfig, SPI_CONST) SpiJobTSBConfig[1] = { { (Spi_JobType)(0), /*Normal Job Id*/ /*DSICR attributes*/ (uint32)(DSPI_DSICR_CS_DPCS0 DSPI_DSICR_TXSS_CLEAR DSPI_DSICR_CID_CLEAR), /*DSICR1 attributes*/ (uint32)(DSPI_DSICR1_CS_DPCS0 DSPI_DSICR1_TSBcnt_31), /*CTAR2 attributes*/ /* External Device Settings */ (uint32)((DSPI_CTAR_CPOL_HIGH /* Clock Polarity (Idle State) */ DSPI_CTAR_CPHA_LEADING /* Clock Phase */ DSPI_CTAR_PCCLK_5 DSPI_CTAR_SCCLK_16 /* TimeCs2Clk: Should=1.0, Is=1, Error=0.0% */ DSPI_CTAR_PASC_5 DSPI_CTAR_SASC_16 /* TimeClk2Cs: Should=1.0, Is=1, Error=0.0% */ DSPI_CTAR_PDT_1 DSPI_CTAR_SDT_512 /* TimeCs2Cs: Should=6.4, Is=6, Error=0.0% */ DSPI_CTAR_PBR_3 DSPI_CTAR_SBR_256 DSPI_CTAR_DBR_0) /* Baudrate: Should=100000.0, Is=104166, Error=4.17% */ (DSPI_CMD_WIDTH_16)) } }; </pre>
------------------------------	--

3.9.2.43 ChangeInDataTransfer

Table 3-95. ChangeInDataTransfer

Description	This bit enables a change in serialization data to initiate DSI frames transfer. Only Continuous, and Change in data initiation control are supported.
Class	Non-Autosar Parameter
Range	true,false
Default	false
Source File	Spi_PbCfg.c

Table continues on the next page...

Table 3-95. ChangelnDataTransfer (continued)

Source Representation	<p>The bold section of the following structure:</p> <pre> CONST(Spi_TSBCConfig, SPI_CONST) SpiJobTSBCConfig[1] = { { (Spi_JobType)(0), /*Normal Job Id*/ /*DSICR attributes*/ (uint32)(DSPI_DSICR_CS_DPCS0 DSPI_DSICR_TXSS_CLEAR DSPI_DSICR_CID_CLEAR), /*DSICR1 attributes*/ (uint32)(DSPI_DSICR1_CS_DPCS0 DSPI_DSICR1_TSBCNT_31), /*CTAR2 attributes*/ /* External Device Settings */ (uint32)((DSPI_CTAR_CPOL_HIGH /* Clock Polarity (Idle State) */ DSPI_CTAR_CPHA_LEADING /* Clock Phase */ DSPI_CTAR_PCCLK_5 DSPI_CTAR_SCCLK_16 /* TimeCs2Clk: Should=1.0, Is=1, Error=0.0% */ DSPI_CTAR_PASC_5 DSPI_CTAR_SASC_16 /* TimeClk2Cs: Should=1.0, Is=1, Error=0.0% */ DSPI_CTAR_PDT_1 DSPI_CTAR_SDT_512 /* TimeCs2Cs: Should=6.4, Is=6, Error=0.0% */ DSPI_CTAR_PBR_3 DSPI_CTAR_SBR_256 DSPI_CTAR_DBR_0) /* Baudrate: Should=100000.0, Is=104166, Error=4.17% */ (DSPI_CMD_WIDTH_16)) } }; </pre>
------------------------------	--

Chapter 4

How to use the same channel in multiple post build Configurations

The Same symbolic name is allowed for the channels which are common across multiple PB configurations by defining a common name if the channelid is same..

Scenario-1:

Let's assume there are 2 post build configurations, CFG1 and CFG2. CFG1 defines 3 channels, CFG2 defines 4 channels. Two of these channels are the same across configurations, let's call them COMCH1 and COMCH2.

So here are the channels defined by the 2 configurations: CFG1: COMCH1, COMCH2, OTHERCH1 CFG2: COMCH1, COMCH2, OTHERCH2, OTHERCH3

This means that CFG1 will use indexes 0, 1, 2, while CFG2 will use indexes 0,1,2,3.

The generated symbolic names for CFG1 and CFG2 will be:

```
#define COMCH1 0
```

```
#define COMCH2 1
```

```
#define OTHERCH1 2
```

```
#define OTHERCH2 2
```

```
#define OTHERCH3 3
```

It allows the upper layer to use same channel symbolic name(s) across multiple configurations.

Scenario-2:

Let's assume there are 4 post build configurations, CFG1, CFG2, CFG3 and CFG4. CFG1 defines 5 channels, CFG2 defines 1 channel, CFG3 defines 3 channels, CFG4 defines 4 channels. One of these channel is same across configurations, let's call it as COMCH1.

So here are the channels defined by the 2 configurations:

CFG1: OTHERCH1, COMCH1, OTHERCH2, OTHERCH3, OTHERCH4

CFG2: OTHERCH5

CFG3: OTHERCH6, COMCH1, OTHERCH7

CFG4: OTHERCH8, COMCH1, OTHERCH9, OTHERCH10

This means that CFG1 will use indexes 0, 1, 2,3,4,CFG2 will use indexes 0, CFG3 will use indexes 0,1,2 and CFG4 will use indexes 0,1,2,3.

The generated symbolic names will be:

```
#define OTHERCH1 0
```

```
#define COMCH1 1
```

```
#define OTHERCH2 2
```

```
#define OTHERCH3 3
```

```
#define OTHERCH4 4
```

```
#define OTHERCH5 0
```

```
#define OTHERCH6 0
```

```
#define OTHERCH7 2
```

```
#define OTHERCH8 0
```

```
#define OTHERCH9 2
```

```
#define OTHERCH10 3
```

Here the same symbolic name is generated for the channel "COMCH1" across the CFG1, CF3 and CFG4. It allows the upper layer to use same channel symbolic name(s) across multiple configurations.

Note: The same channelid should be used across multiple PB configurations in order to use same symbolic name across multiple PB configurations. All channel attributes shall be identical for cloned channels.,

Chapter 5

DEM events reported by SPI driver

5.1 DEM events

Production errors (DEM) reported by SPI driver are explained in below table.

Table 5-1. Production Errors (DEM) reported by SPI driver

Function in which DEM error is reported	DEM event ID	Conditions that generate this DEM event	fatal HW error, i.e. ECU is defective and needs to be replaced ?
Spi_LLD_SyncTransmit_Fast()	SPI_E_TIMEOUT	Transmission command has not been accepted	YES
Spi_LLD_SyncTransmit()	SPI_E_TIMEOUT	Transmission command has not been accepted	YES

How to Reach Us:

Home Page:

www.freescale.com

Web Support:

<http://www.freescale.com/support>

USA/Europe or Locations Not Listed:

Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:

Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:

Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:

Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductors products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claims alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-complaint and/or non-Pb-free counterparts. For further information, see <http://www.freescale.com> or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to <http://www.freescale.com/epp>.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2011 Freescale Semiconductor, Inc.