User Manual

for MPC574XG SPI Driver

Document Number: UM35SPIASR4.2 Rev0002R1.0.0
Rev. 1.0

Contents

Section number **Title** **Page**

Chapter 1
Revision History

Chapter 2
Introduction

Chapter 3
Driver

Chapter 4
Tresos Configuration Plug-in

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0	17/02/2017	Nghiem Toan	Calypso ASR4.2 RTM 1.0.0 Release

Chapter 2
Introduction

This User Manual describes NXP Semiconductor AUTOSAR Serial Peripheral Interface (SPI) for MPC574XG .

AUTOSAR SPI driver configuration parameters and deviations from the specification are described in SPI Driver chapter of this document. AUTOSAR SPI driver requirements and APIs are described in the AUTOSAR SPI driver software specification document.

2.1 Supported Derivatives

The software described in this document is intented to be used with the following microcontroller devices of NXP Semiconductor .

Table 2-1. MPC574XG Derivatives

NXP Semiconductor	MPC5748G_LQFP176, MPC5748G_MAPBGA256, MPC5748G_MAPBGA324, MPC5747G_LQFP176, MPC5747G_MAPBGA256, MPC5747G_MAPBGA324, MPC5746G_LQFP176, MPC5746G_MAPBGA256, MPC5746G_MAPBGA324, MPC5748C_LQFP176, MPC5748C_MAPBGA256, MPC5748C_MAPBGA324, MPC5747C_LQFP176, MPC5747C_MAPBGA256, MPC5747C_MAPBGA324, MPC5746C_LQFP176, MPC5746C_MAPBGA256, MPC5746C_MAPBGA324, MPC5746C_MAPBGA100, MPC5745C_LQFP176, MPC5745C_MAPBGA256, MPC5745C_MAPBGA100, MPC5744C_LQFP176, MPC5744C_MAPBGA256, MPC5744C_MAPBGA100,

Table 2-1. MPC574XG Derivatives

	MPC5746B_LQFP176, MPC5746B_MAPBGA256, MPC5746B_MAPBGA100, MPC5744B_LQFP176, MPC5744B_MAPBGA256, MPC5744B_MAPBGA100, MPC5745B_LQFP176, MPC5745B_MAPBGA256, MPC5745B_MAPBGA100

All of the above microcontroller devices are collectively named as MPC574XG .

2.2 Overview

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

AUTOSAR

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

2.3 About this Manual

This Technical Reference employs the following typographical conventions:

Boldface 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.

Notes and warnings are shown as below:

Note

This is a note.

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
BSW	Basic Software
N/A	Not Applicable
ISR	Interrupt Service Routine
OS	Operating System
MCU	Microcontroller Unit
GUI	Graphical User Interface
PB Variant	Post Build Variant
PC Variant	Pre Compile Variant
LT Variant	Link Time Variant

2.5 Reference List

Table 2-3. Reference List

#	Title	Version
1	AUTOSAR 4.2 Rev0002SPI Driver Software Specification Document.	R4.2 Rev 002
2	MPC5748G Reference Manual	Rev. 5, 12/2016
3	MPC5746C Reference Manual	Rev. 4, 12/2016
4	MPC5748G_1N81M_Rev.2 (official document) (1N81M)	Jun-16
5	MPC5748G_1N81M_0N78S_Comparison_Summary_v2_0 (internal document) (1N81M, 0N78S)	31.10.2016
6	MPC5746C_1N06M_Rev.4 (official document) (1N06M)	Jul-16
7	MPC5746C_cut1.1_cut2.0_cut2.1_comparison_v0 (internal document) (1N06M, 0N84S, 1N84S)	14-Sep-16
8	C3M_cut2.1_new_errata_20170113 (internal document) (1N84S)	13-Jan-17

Chapter 3
Driver

3.1 Requirements

Requirements for this driver are detailed in the AUTOSAR 4.2 Rev0002SPI Driver Software Specification document (See Table Reference List).

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 specification 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.

Notification usage

To be able to use the SPI driver with DMA functionality, the following function need to be set as notification for the DMA channels used: Spi_Dspi_IsrRxDma_DSPI_X, where X is the DSPI unit used (eg: Spi_Dspi_IsrRxDma_DSPI_0 if DSPI0 is used).

Interrupt request usage

Every interrupt is guarded by #ifdef definitions that specify if the corresponded 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

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 SpiConf_SpiSequence_<SpiSequenceName> ((Spi_SequenceType)<SpiSequenceID>)

- for jobs:

#define SpiConf_SpiJob_<SpiJobName> ((Spi_JobType)<SpiJobID>)

- for channels:

#define SpiConf_SpiChannel_<SpiChannelName> ((Spi_ChannelType)<SpiChannelID>)

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

3.3 Hardware Resources

The hardware configured by the Spi driver is the same between derivatives.

3.4 Deviation from Requirements

Table 3.1 Deviations Status Column Description identifies the AUTOSAR requirements that are not fully implemented, implemented differently, or out of scope for the SPI driver. Table Table 3-2 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
SWS_Spi_00270	N/S	In case call end notification function and rescheduling are fully done by software, the order between these shall be first scheduling and then the call of end notification function executed.	Job and sequences notifications are performed before the scheduling of the next job (contrary to the recommendation given by SPI270) . 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.
ECUC_Spi_00238	N/S	If SpiHwUnitSynchronous is set to "SYNCHRONOUS", the SpiJob uses its containing SpiDriver in a synchronous manner. If it is set to "ASYNCHRONOUS", it uses the driver in an asynchronous way. If the parameter is not set, the SpiChannel uses the driver also in an asynchronous way.	The SpiHwUnitSynchronous parameter is defined at Job level in Autosar. It should be defined at HwUnit level. This paramater is added for AutoSAR compatibility purpose and this parameter is not used.The parameter SpiPhyUnitSync located in SpiPhyUnit tab is used for the same purpose.
ECUC_Spi_00239	N/S	When the Chip select handling is enabled (see SpiEnableCs), then this parameter specifies if the chip select is handled automatically by Peripheral HW engine or via general purpose IO by Spi driver.	If user selects paraeter CS_VIA_GPIO, the user has to use SpiJobStartNotification' & 'SpiJobEndNotification' to toggle the CS (chip select pin) using DIO drivers .
SWS_Spi_00195	N/S	SPI Handler/driver shall be able to detect the error SPI_E_HARDWARE_ERROR when an hardware error occur during asynchronous or	SPI_E_HARDWARE_ERROR is not supported for Spi_Async_Transmit() Function. It is supported for Spi_Sync_Transmit() function.

Table continues on the next page...

Table 3-2. SPI Deviations Table (continued)

Requirement	Status	Description	Notes
		synchronous transmit. Please see also SWS_Spi_00267 and SWS_Spi_00384.	To implement this requirement a timer should be set to that estimated value and if the timer expires, then it can be assumed that a hardware error occurred. This would add a dependency of Gpt.
SWS_Spi_00383	N/S	Error Name: SPI_E_HARDWARE_ERROR Short Description: An hardware error occurred during asynchronous or synchronous SPI transmit. Long Description: This Extended Production Error shall be issued when any error bit inside the SPI hardware transmit status register is raised Detection Criteria: Fail The SPI transmit status register information shall be reported to DEM as Dem_ReportErrorStatus (SPI_E_HARDWARE_ERROR, DEM_EVENT_STATUS_FAILED) when any error bit inside the SPI transmit status register is set. (SWS_Spi_00385) Pass The SPI transmit status register information shall be reported to DEM as Dem_ReportErrorStatus (SPI_E_HARDWARE_ERROR, DEM_EVENT_STATUS_PASSED) when no error bit inside the SPI transmit status register is set. (SWS_Spi_00386)	SPI_E_HARDWARE_ERROR is not supported for Spi_Async_Transmit() Function. It is supported for Spi_Sync_Transmit() function. To implement this requirement a timer should be set to that estimated value and if the timer expires, then it can be assumed that a hardware error occurred. This would add a dependency of Gpt.
SWS_Spi_00385	N/S	When any error bit inside the SPI transmit status register is set, the SPI transmit status register information shall be reported to DEM as Dem_ReportErrorStatus (SPI_E_HARDWARE_ERROR, DEM_EVENT_STATUS_FAILED)	SPI_E_HARDWARE_ERROR is not supported for Spi_Async_Transmit() Function. It is supported for Spi_Sync_Transmit() function. To implement this requirement a timer should be set to that estimated value and if the timer expires, then it can be assumed that a hardware error occurred. This would add a dependency of Gpt.
SWS_Spi_00386	N/S	When no error bit inside the SPI transmit status register is set, the SPI transmit status register information shall be reported to DEM as Dem_ReportErrorStatus (SPI_E_HARDWARE_ERROR, DEM_EVENT_STATUS_PASSED)	SPI_E_HARDWARE_ERROR is not supported for Spi_Async_Transmit() Function. It is supported for Spi_Sync_Transmit() function. To implement this requirement a timer should be set to that estimated value and if the timer expires, then it can be assumed that a hardware error occurred. This would add a dependency of Gpt.
SWS_Spi_00293	N/S	When the function Spi_AsyncTransmit is called, the SPI Handler/Driver shall handle the Job results. Result shall be SPI_JOB_FAILED when the transmission of Jobs is failed.	The Spi_AsyncTransmit can only schedule Jobs to be sent. So the function itself cannot detect if a job is failed.

3.5 Driver limitations

- Driver does not support use parity control.

- Driver does not support expandable of Chip Selects with external demultiplexer.

- Driver does not support Modified SPI transfer formats.

- Interrupt conditions do not use by driver:

  - CMD FIFO is not full (CMDFFF)
  - End of Queue reached (EOQF)
  - Transfers due from current command frame complete (CMDTCF)
  - Transfer of current SPI frame Complete (SPITCF)
  - Attempt to transmit with an empty Transmit FIFO (TFUF)
  - Frame received while Receive FIFO is full (RFOF)
  - SPI Parity Error (SPEF)
  - Data present in TX FIFO while CMD FIFO is empty (TFIWF)

- Driver does not support the following features in TSB and Interleaved TSB modes:

  - The transfer initiation condition is edge sensitive hardware trigger
  - Frames from SPI and DSI are identifiable by a bit transmitted at the start of each frame
  - Separate interrupts for frame completion from DSI

3.6 Driver usage and configuration tips

3.6.1 How to the configuration in multiple post build variants.

To use multiple post build variants, the configurations need to have the same symbolic name. So, all Names and IDs of Sequences, Jobs, Channels must the same between all post build variants. Names and Index of ExternalDevices must the same between all post build variants.

Scenario:

Let’s assume there are 2 post build variant configurations, VS_0 and VS_1.

- VS_0 defines 2 Sequences, 2 Jobs, 2 Channels and 2 ExternalDevices:

  - Name of sequence is SEQ_DSPI0_1J_C0_1 and SpiSequenceId is 0.
  - Name of job is JOB_DSPI0_C0_1 and SpiJobId is 0.
  - Name of ExternalDevice is DEV_EXP_100K_LEAD_0 and Index is 0.
  - Name of channel is CH_EB_10K and SpiChannelId is 0.

- Name of sequence is SEQ_DSPI1_1J_C0_2 and SpiSequenceId is 1.
- Name of job is JOB_DSPI1_C0_2 and SpiJobId is 1.
- Name of ExternalDevice is DEV_EXP_100K_LEAD_1 and Index is 1.
- Name of channel is CH_EB_1K and SpiChannelId is 1.

So, VS_1 must has configuration the same Names, IDs and Indexs with VS_0 for Sequences, Jobs, Channels and ExternalDevices.

- VS_1 defines 2 Sequences, 2 Jobs, 2 Channels and 2 ExternalDevices:

- Name of sequence is SEQ_DSPI0_1J_C0_1 and SpiSequenceId is 0.
- Name of job is JOB_DSPI0_C0_1 and SpiJobId is 0.
- Name of ExternalDevice is DEV_EXP_100K_LEAD_0 and Index is 0.
- Name of channel is CH_EB_10K and SpiChannelId is 0.

- Name of sequence is SEQ_DSPI1_1J_C0_2 and SpiSequenceId is 1.
- Name of job is JOB_DSPI1_C0_2 and SpiJobId is 1.
- Name of ExternalDevice is DEV_EXP_100K_LEAD_1 and Index is 1.
- Name of channel is CH_EB_1K and SpiChannelId is 1.

The generated symbolic names for VS_0 and VS_1 will be:

#define SpiConf_SpiSequence_SEQ_DSPI0_1J_C0_1 0

#define SpiConf_SpiSequence_SEQ_DSPI1_1J_C0_2 1

#define SpiConf_SpiJob_JOB_DSPI0_C0_1 0

#define SpiConf_SpiJob_JOB_DSPI1_C0_2 1

#define SpiConf_SpiChannel_CH_EB_10K 0

#define SpiConf_SpiChannel_CH_EB_1K 1

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

3.7 Runtime Errors

The driver generates the following DEM errors at runtime.

Table 3-3. Runtime Errors

Function	Error Code	Condition triggering the error
Spi_<IPV_Name>_SyncTransmit_Fast	Spi_E_Hardware_ErrorCfg	The SPI driver cannot transmit complete or receive complete one frame in the allocated time defined by "SPI_TIMEOUT_COUNTER" parameter in configuration. Baud rate of HW might be low speed or timeout value to short. Timeout occured.
Spi_<IPV_Name>_SyncTransmit	Spi_E_Hardware_ErrorCfg	The SPI driver cannot transmit complete or receive complete one frame in the allocated time defined by "SPI_TIMEOUT_COUNTER" parameter in configuration. Baud rate of HW might be low speed or timeout value to short. Timeout occured.

3.8 Software specification

The following sections contains driver software specifications.

3.8.1 Define Reference

Constants supported by the driver are as per AUTOSAR SPI Driver software specification Version 4.2 Rev0002 .

3.8.1.1 Define SPI_ASYNCTRANSMIT_ID

API service ID for SPI async transmit function.

Details:

Parameters used when raising an error/exception.

Table 3-4. Define SPI_ASYNCTRANSMIT_ID Description

Name	SPI_ASYNCTRANSMIT_ID
Initializer	((uint8) 0x03u)

3.8.1.2 Define SPI_CANCEL_ID

API service ID for SPI cancel function.

Details:

Parameters used when raising an error/exception.

Table 3-5. Define SPI_CANCEL_ID Description

Name	SPI_CANCEL_ID
Initializer	((uint8) 0x0Cu)

3.8.1.3 Define SPI_DEINIT_ID

API service ID for SPI DeInit function.

Details:

Parameters used when raising an error/exception.

Table 3-6. Define SPI_DEINIT_ID Description

Name	SPI_DEINIT_ID
Initializer	((uint8) 0x01u)

3.8.1.4 Define SPI_E_ALREADY_INITIALIZED

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

Table 3-7. Define SPI_E_ALREADY_INITIALIZED Description

Name	SPI_E_ALREADY_INITIALIZED
Initializer	((uint8)0x4Au)

3.8.1.5 Define SPI_E_CONFIG_OUT_OF_RANGE

The number of sequences, jobs or channels exceeds precompile time sizes.

Details:

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.

Table 3-8. Define SPI_E_CONFIG_OUT_OF_RANGE Description

Name	SPI_E_CONFIG_OUT_OF_RANGE
Initializer	((uint8)0x5Au)

3.8.1.6 Define SPI_E_JOB_EMPTY

The number of channel in job is zero.

Details:

The number of channel in the job configuration is zero. Verification before transfer sequence.

Table 3-9. Define SPI_E_JOB_EMPTY Description

Name	SPI_E_JOB_EMPTY
Initializer	((uint8)0x5Du)

3.8.1.7 Define SPI_E_PARAM_CHANNEL

API service called with wrong parameter of channel.

Details:

The parameter in channel configuration is wrong as: Channel > Spi_Max_Channel, wrong buffer type, DataBufferPointer of internal buffer is NULL. Verification by function Spi_WriteIB(), Spi_ReadIB(), Spi_SetupEB().

Table 3-10. Define SPI_E_PARAM_CHANNEL Description

Name	SPI_E_PARAM_CHANNEL
Initializer	((uint8)0x0Au)

3.8.1.8 Define SPI_E_PARAM_EB_UNIT

When a sequence contains uninitialized external buffers.

Details:

All external buffer in sequence must initialization before transfer. Verification before transfer sequence by function Spi_AsyncTransmit(), Spi_SyncTransmit().

Table 3-11. Define SPI_E_PARAM_EB_UNIT Description

Name	SPI_E_PARAM_EB_UNIT
Initializer	((uint8)0x5Bu)

3.8.1.9 Define SPI_E_PARAM_JOB

API service called with wrong parameter of job.

Details:

The parameter in job configuration is wrong as: Job > Spi_Max_Job. Verification by function Spi_GetJobResult().

Table 3-12. Define SPI_E_PARAM_JOB Description

Name	SPI_E_PARAM_JOB
Initializer	((uint8)0x0Bu)

3.8.1.10 Define SPI_E_PARAM_LENGTH

API service called with wrong parameter of external buffer.

Details:

The length of external buffer is wrong as: Length=0, Length>Max_Length. Verification by function Spi_SetupEB().

Table 3-13. Define SPI_E_PARAM_LENGTH Description

Name	SPI_E_PARAM_LENGTH
Initializer	((uint8)0x0Du)

3.8.1.11 Define SPI_E_PARAM_POINTER

If the parameter versioninfo is NULL_PTR.

Details:

Indicate version information of SPI driver is NULL_PTR. Verification by function Spi_GetVersionInfo().

Table 3-14. Define SPI_E_PARAM_POINTER Description

Name	SPI_E_PARAM_POINTER
Initializer	((uint8)0x10u)

3.8.1.12 Define SPI_E_PARAM_SEQ

API service called with wrong parameter in sequence configuration.

Details:

The parameter in sequence configuration is wrong as: Sequence>Spi_Max_Sequence. Verification before transfer sequence by function Spi_AsyncTransmit(), Spi_GetSequenceResult(), Spi_SyncTransmit(), Spi_Cancel().

Table 3-15. Define SPI_E_PARAM_SEQ Description

Name	SPI_E_PARAM_SEQ
Initializer	((uint8)0x0Cu)

3.8.1.13 Define SPI_E_PARAM_UNIT

API service called with wrong parameter of HW unit.

Details:

The parameter in HW unit configuration is wrong as: transfer mode, HWUnit > SPI_MAX_HWUNIT. Verification by function Spi_AsyncTransmit(), Spi_SyncTransmit(), Spi_GetHWUnitStatus(), Spi_SetHWUnitAsyncMode().

Table 3-16. Define SPI_E_PARAM_UNIT Description

Name	SPI_E_PARAM_UNIT
Initializer	((uint8)0x0Eu)

3.8.1.14 Define SPI_E_SEQ_EMPTY

No job in sequence.

Details:

The number of job in sequence configuration is zero. Verification before transfer sequence by function Spi_AsyncTransmit().

Table 3-17. Define SPI_E_SEQ_EMPTY Description

Name	SPI_E_SEQ_EMPTY
Initializer	((uint8)0x5Cu)

3.8.1.15 Define SPI_E_SEQ_IN_PROCESS

Synchronous transmission service called at wrong time.

Details:

Indicate HW unit is already running. Verification by function Spi_SyncTransmit().

Table 3-18. Define SPI_E_SEQ_IN_PROCESS Description

Name	SPI_E_SEQ_IN_PROCESS
Initializer	((uint8)0x3Au)

3.8.1.16 Define SPI_E_SEQ_PENDING

Services called in a wrong sequence.

Details:

Indicate sequence status is pending or used HW unit is busy.

Table 3-19. Define SPI_E_SEQ_PENDING Description

Name	SPI_E_SEQ_PENDING
Initializer	((uint8)0x2Au)

3.8.1.17 Define SPI_E_UNINIT

API service used without module initialization.

Details:

Indicate SPI driver is uninitialized .

Table 3-20. Define SPI_E_UNINIT Description

Name	SPI_E_UNINIT
Initializer	((uint8)0x1Au)

3.8.1.18 Define SPI_GETHWUNITSTATUS_ID

API service ID for SPI get hwunit status function.

Details:

Parameters used when raising an error/exception.

Table 3-21. Define SPI_GETHWUNITSTATUS_ID Description

Name	SPI_GETHWUNITSTATUS_ID
Initializer	((uint8) 0x0Bu)

3.8.1.19 Define SPI_GETJOBRESULT_ID

API service ID for SPI get job result function.

Details:

Parameters used when raising an error/exception.

Table 3-22. Define SPI_GETJOBRESULT_ID Description

Name	SPI_GETJOBRESULT_ID
Initializer	((uint8) 0x07u)

3.8.1.20 Define SPI_GETSEQUENCERESULT_ID

API service ID for SPI get sequence result function.

Details:

Parameters used when raising an error/exception.

Table 3-23. Define SPI_GETSEQUENCERESULT_ID Description

Name	SPI_GETSEQUENCERESULT_ID
Initializer	((uint8) 0x08u)

3.8.1.21 Define SPI_GETSTATUS_ID

API service ID for SPI get status function.

Details:

Parameters used when raising an error/exception.

Table 3-24. Define SPI_GETSTATUS_ID Description

Name	SPI_GETSTATUS_ID
Initializer	((uint8) 0x06u)

3.8.1.22 Define SPI_GETVERSIONINFO_ID

API service ID for SPI get version info function.

Details:

Parameters used when raising an error/exception.

Table 3-25. Define SPI_GETVERSIONINFO_ID Description

Name	SPI_GETVERSIONINFO_ID
Initializer	((uint8) 0x09u)

3.8.1.23 Define SPI_INIT_ID

API service ID for SPI Init function.

Details:

Parameters used when raising an error/exception.

Table 3-26. Define SPI_INIT_ID Description

Name	SPI_INIT_ID
Initializer	((uint8) 0x00u)

3.8.1.24 Define SPI_JOB_PRIORITY_LEVELS_COUNT

The number of allowed job priority levels (0..3).

Details:

The Priority has to be sint8.

Table 3-27. Define SPI_JOB_PRIORITY_LEVELS_COUNT Description

Name	SPI_JOB_PRIORITY_LEVELS_COUNT
Initializer	(4)

3.8.1.25 Define SPI_MAINFUNCTION_HANDLING_ID

API service ID for SPI main function.

Details:

Parameters used when raising an error/exception

Table 3-28. Define SPI_MAINFUNCTION_HANDLING_ID Description

Name	SPI_MAINFUNCTION_HANDLING_ID
Initializer	((uint8)0x10u)

3.8.1.26 Define SPI_READIB_ID

API service ID for SPI read IB function.

Details:

Parameters used when raising an error/exception.

Table 3-29. Define SPI_READIB_ID Description

Name	SPI_READIB_ID
Initializer	((uint8) 0x04u)

3.8.1.27 Define SPI_SETASYNCMODE_ID

API service ID for SPI set async mode function.

Details:

Parameters used when raising an error/exception.

Table 3-30. Define SPI_SETASYNCMODE_ID Description

Name	SPI_SETASYNCMODE_ID
Initializer	((uint8) 0x0Du)

3.8.1.28 Define SPI_SETCLOCKMODE_ID

API service ID for SPI Set Clock Mode.

Details:

Parameters used when raising an error/exception.

Table 3-31. Define SPI_SETCLOCKMODE_ID Description

Name	SPI_SETCLOCKMODE_ID
Initializer	((uint8)0x81u)

3.8.1.29 Define SPI_SETHWUNITASYNCMODE_ID

API service ID for SPI set HW Unit async mode.

Details:

Parameters used when raising an error/exception.

Table 3-32. Define SPI_SETHWUNITASYNCMODE_ID Description

Name	SPI_SETHWUNITASYNCMODE_ID
Initializer	((uint8)0x80u)

3.8.1.30 Define SPI_SETUPEB_ID

API service ID for SPI setup EB function.

Details:

Parameters used when raising an error/exception.

Table 3-33. Define SPI_SETUPEB_ID Description

Name	SPI_SETUPEB_ID
Initializer	((uint8) 0x05u)

3.8.1.31 Define SPI_SYNCTRANSMIT_ID

API service ID for SPI sync transmit function.

Details:

Parameters used when raising an error/exception.

Table 3-34. Define SPI_SYNCTRANSMIT_ID Description

Name	SPI_SYNCTRANSMIT_ID
Initializer	((uint8) 0x0Au)

3.8.1.32 Define SPI_WRITEIB_ID

API service ID for SPI write IB function.

Details:

Parameters used when raising an error/exception.

Table 3-35. Define SPI_WRITEIB_ID Description

Name	SPI_WRITEIB_ID
Initializer	((uint8) 0x02u)

3.8.1.33 Define SPI_ALLOW_BIGSIZE_COLLECTIONS

If enabled, allows to configure more than 256 sequences, jobs and channels.

Table 3-36. Define SPI_ALLOW_BIGSIZE_COLLECTIONS Description

Name	SPI_ALLOW_BIGSIZE_COLLECTIONS
Initializer	(STD_OFF)

3.8.1.34 Define SPI_CANCEL_API

Switches the Spi_Cancel function ON or OFF.

Details:

Switches the Spi_Cancel function ON or OFF. (see chapter 8.3.13)

Table 3-37. Define SPI_CANCEL_API Description

Name	SPI_CANCEL_API
Initializer	(STD_ON)

3.8.1.35 Define SPI_CHANNEL_BUFFERS_ALLOWED

Selects the SPI Handler/Driver Channel Buffers usage allowed and delivered.

Details:

Selects the SPI Handler/Driver Channel Buffers usage allowed and delivered. (see chapter 7.2.1)

Table 3-38. Define SPI_CHANNEL_BUFFERS_ALLOWED Description

Name	SPI_CHANNEL_BUFFERS_ALLOWED
Initializer	(USAGE2)

3.8.1.36 Define SPI_CONFIG_VARIANT

Defines the use of Pre-Compile(PC) support.

Details:

VARIANT-PRE-COMPILE: Only parameters with "Pre-compile time" configuration are allowed in this variant.

Table 3-39. Define SPI_CONFIG_VARIANT Description

Name	SPI_CONFIG_VARIANT
Initializer	(SPI_VARIANT_PRECOMPILE)

3.8.1.37 Define SPI_DEV_ERROR_DETECT

Switches the Development Error functionality ON or OFF.

Details:

Switches the Development Error Detection and Notification ON or OFF.

Table 3-40. Define SPI_DEV_ERROR_DETECT Description

Name	SPI_DEV_ERROR_DETECT
Initializer	(STD_ON)

3.8.1.38 Define SPI_DISABLE_DEM_REPORT_ERROR_STATUS

Switches the Production Error Detection and Notification OFF.

Table 3-41. Define SPI_DISABLE_DEM_REPORT_ERROR_STATUS Description

Name	SPI_DISABLE_DEM_REPORT_ERROR_STATUS
Initializer	(STD_ON)

3.8.1.39 Define SPI_DMA_USED

Defines if transfers are made using DMA or FIFO.

Details:

Defines if transfers are made using DMA or FIFO.

Table 3-42. Define SPI_DMA_USED Description

Name	SPI_DMA_USED
Initializer	(STD_ON)

3.8.1.40 Define SPI_DUAL_CLOCK_MODE

If enabled, allows dual MCU clock configuration settings.

Details:

If enabled, allows dual MCU clock configuration settings.

Table 3-43. Define SPI_DUAL_CLOCK_MODE Description

Name	SPI_DUAL_CLOCK_MODE
Initializer	(STD_OFF)

3.8.1.41 Define SPI_HW_STATUS_API

Switches the Spi_GetHWUnitStatus function ON or OFF.

Details:

Switches the Spi_GetHWUnitStatus function ON or OFF.

Table 3-44. Define SPI_HW_STATUS_API Description

Name	SPI_HW_STATUS_API
Initializer	(STD_ON)

3.8.1.42 Define SPI_HWUNIT_ASYNC_MODE

If enabled, the asyncronous operation mode (POLLING or INTERRUPT).

Details:

If enabled, the asyncronous operation mode (POLLING or INTERRUPT) can be defined independently for each HWUnit using`Spi_SetHWUnitAsyncMode()`.

Table 3-45. Define SPI_HWUNIT_ASYNC_MODE Description

Name	SPI_HWUNIT_ASYNC_MODE
Initializer	(STD_ON)

3.8.1.43 Define SPI_INTERRUPTIBLE_SEQ_ALLOWED

Switches the Interruptible Sequences handling functionality ON or OFF.

Details:

This parameter depends on SPI_LEVEL_DELIVERED value. It is only used for SPI_LEVEL_DELIVERED configured to 1 or 2.

Table 3-46. Define SPI_INTERRUPTIBLE_SEQ_ALLOWED Description

Name	SPI_INTERRUPTIBLE_SEQ_ALLOWED
Initializer	(STD_ON)

3.8.1.44 Define SPI_LEVEL_DELIVERED

Selects the SPI Handler/Driver level of scalable functionality.

Details:

Selects the SPI Handler/Driver level of scalable functionality that is available and delivered. (see chapter 7.1)

Table 3-47. Define SPI_LEVEL_DELIVERED Description

Name	SPI_LEVEL_DELIVERED
Initializer	(LEVEL2)

3.8.1.45 Define SPI_OPTIMIZE_ONE_JOB_SEQUENCES

Defines if Spi driver optimization for sequences having only one job is activated or not.

Details:

Defines if Spi driver optimization for sequences having only one job is activated or not. If activated, additional RAM memory is required for internal data caching.

Table 3-48. Define SPI_OPTIMIZE_ONE_JOB_SEQUENCES Description

Name	SPI_OPTIMIZE_ONE_JOB_SEQUENCES
Initializer	(STD_ON)

3.8.1.46 Define SPI_OPTIMIZED_CHANNEL_BUFFER_SIZE

Define the size of channel cached data buffer.

Details:

Define the size of channel cached data buffer for sequences having only one job.

Table 3-49. Define SPI_OPTIMIZED_CHANNEL_BUFFER_SIZE Description

Name	SPI_OPTIMIZED_CHANNEL_BUFFER_SIZE
Initializer	((`Spi_ChannelType`)50)

3.8.1.47 Define SPI_OPTIMIZED_SEQ_BUFFER_SIZE

Define the size of sequence cached data buffer.

Details:

Define the size of sequence cached data buffer for sequences having only one job.

Table 3-50. Define SPI_OPTIMIZED_SEQ_BUFFER_SIZE Description

Name	SPI_OPTIMIZED_SEQ_BUFFER_SIZE
Initializer	((`Spi_SequenceType`)44)

3.8.1.48 Define SPI_SUPPORT_CONCURRENT_SYNC_TRANSMIT

Allow simultaneous calls to`Spi_SyncTransmit()` for different threads.

Details:

Two concurrent calls to`Spi_SyncTransmit()` will be allowed only if the related sequences do not share HW units.

Table 3-51. Define SPI_SUPPORT_CONCURRENT_SYNC_TRANSMIT Description

Name	SPI_SUPPORT_CONCURRENT_SYNC_TRANSMIT
Initializer	(STD_OFF)

3.8.1.49 Define SPI_TIMEOUT_COUNTER

Defines the "Number of Loops" timeout.

Details:

Defines the "Number of Loops" timeout used by Spi_SyncTransmit and Spi_AsyncTransmit functions during the wait on TX/RX transmission to complete one frame. One timeout unit means that no TX or RX was executed(the IF statements are returning FALSE).

Table 3-52. Define SPI_TIMEOUT_COUNTER Description

Name	SPI_TIMEOUT_COUNTER
Initializer	((uint32)(((SpiTransmitTimeout * CoreFrequency) div 1000000) / SPI_WAIT_LOOP_TICKS))

CoreFrequency is value reference to CPU Clock by SpiCPUClockRef node.

3.8.1.50 Define SPI_VERSION_INFO_API

Switches the Version Information API functionality ON or OFF.

Details:

Switches the Spi_GetVersionInfo function ON or OFF.

Table 3-53. Define SPI_VERSION_INFO_API Description

Name	SPI_VERSION_INFO_API
Initializer	(STD_ON)

3.8.1.51 Define SPI_WAIT_LOOP_TICKS

Number of CPU clock cycles consumed by a wait loop during the wait for TX/RX transmission to complete one frame in Spi_SyncTransmit.

Details:

This value is set to the minimum measure retrieved for GHS, DIAB and CW compilers, with all optimizations activated.

Table 3-54. Define SPI_WAIT_LOOP_TICKS Description

Name	SPI_WAIT_LOOP_TICKS
Initializer	23u

3.8.1.52 Define SPI_PRECOMPILE_SUPPORT

Define precompile support.

Details:

Define precompile support if VariantPreCompile or VariantLinkTime is selected and number of variant lower or equal 1.

Table 3-55. Define SPI_PRECOMPILE_SUPPORT Description

Name	SPI_PRECOMPILE_SUPPORT
Initializer	(STD_OFF)

3.8.2 Enum Reference

Enumeration of all constants supported by the driver are as per AUTOSAR SPI Driver software specification Version 4.2 Rev0002 .

3.8.2.1 Enumeration Spi_AsyncModeType

Specifies the asynchronous mechanism mode for SPI busses handled asynchronously in Level 2.

Details:

if (LEVEL2 == SPI_LEVEL_DELIVERED) Specifies the asynchronous mechanism mode for SPI busses handled asynchronously in LEVEL 2. SPI150: This type is available or not according to the pre compile time parameter: SPI_LEVEL_DELIVERED. This is only relevant for LEVEL 2.

Note

The default value of AsyncModeType is SPI_POLLING_MODE. In order to use the interrupt for asynchronous behavior, the user must call the function Spi_SetAsyncMode(AsyncMode). Please see the chapter for the description of the function.

Table 3-56. Enumeration Spi_AsyncModeType Values

Name	Initializer	Description
SPI_POLLING_MODE	0	
SPI_INTERRUPT_MODE	1	

3.8.2.2 Enumeration Spi_BufferType

The enumeration containg the designated values for buffer types (internal or external).

Table 3-57. Enumeration Spi_BufferType Values

Name	Initializer	Description
IB	0	
EB		

3.8.2.3 Enumeration Spi_JobResultType

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

Table 3-58. Enumeration Spi_JobResultType Values

Name	Initializer	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.
SPI_JOB_QUEUED		An asynchronous transmit Job has been accepted, while actual transmission for this Job has not started yet.

3.8.2.4 Enumeration Spi_ModeType

Table 3-59. Enumeration Spi_ModeType Values

Name	Initializer	Description
SPI_MASTER	0	SPI Hardware selected as MASTER.
SPI_SLAVE		SPI Hardware selected as SLAVE.

3.8.2.5 Enumeration Spi_SeqResultType

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

Table 3-60. Enumeration Spi_SeqResultType Values

Name	Initializer	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.

3.8.2.6 Enumeration Spi_StatusType

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

Table 3-61. Enumeration Spi_StatusType Values

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

3.8.3 Function Reference

Functions of all functions supported by the driver are as per AUTOSAR SPI Driver software specification Version 4.2 Rev0002 .

3.8.3.1 Function Spi_AsyncTransmit

This function triggers the asynchronous transmission for the given sequence.

Details:

This function triggers the asynchronous transmission for the given sequence.

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

Return: Std_ReturnType.

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

Prototype: `Std_ReturnType Spi_AsyncTransmit(Spi_SequenceType Sequence);`

Table 3-62. Spi_AsyncTransmit Arguments

Type	Name	Direction	Description
`Spi_SequenceType`	Sequence	**input**	Sequence ID.

Table 3-63. Spi_AsyncTransmit Return Values

Name	Description
E_OK	Transmission command has been accepted.
E_NOT_OK	Transmission command has not been accepted.

3.8.3.2 Function Spi_Cancel

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

Details:

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()` shall raise 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.

Prototype: `void Spi_Cancel(Spi_SequenceType Sequence);`

Table 3-64. Spi_Cancel Arguments

Type	Name	Direction	Description
`Spi_SequenceType`	Sequence	**input**	Sequence ID.

3.8.3.3 Function Spi_DeInit

This function de-initializes the SPI driver.

Details:

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

- Service ID: 0x01
- Sync/Async: Synchronous
- Reentrancy: Non-Reentrant

Return: Std_ReturnType.

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

Prototype: `Std_ReturnType Spi_DeInit(void);`

Table 3-65. Spi_DeInit Return Values

Name	Description
E_OK	De-initialisation command has been accepted.
E_NOT_OK	De-initialisation command has not been accepted.

3.8.3.4 Function Spi_GetAsyncStatus

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

Details:

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

Return: Spi_StatusType.

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL2.

Prototype: `LOCAL_INLINESpi_StatusType Spi_GetAsyncStatus(void);`

Table 3-66. Spi_GetAsyncStatus Return Values

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

3.8.3.5 Function Spi_GetHWUnitStatus

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

Details:

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

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

Return: Spi_StatusType.

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

Prototype: `Spi_StatusType Spi_GetHWUnitStatus(Spi_HWUnitType HWUnit);`

Table 3-67. Spi_GetHWUnitStatus Arguments

Type	Name	Direction	Description
`Spi_HWUnitType`	HWUnit	**input**	The HW peripheral for which we need the status.

Table 3-68. Spi_GetHWUnitStatus Return Values

Name	Description
SPI_UNINIT	The peripheral is un-initialized.
SPI_IDLE	The peripheral is in idle state.
SPI_BUSY	The peripheral is busy.

3.8.3.6 Function Spi_GetJobResult

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

Details:

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

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

Return: Spi_JobResultType.

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

Prototype: `Spi_JobResultType Spi_GetJobResult(Spi_JobType Job);`

Table 3-69. Spi_GetJobResult Arguments

Type	Name	Direction	Description
`Spi_JobType`	Job	**input**	Job ID.

Table 3-70. Spi_GetJobResult Return Values

Name	Description
SPI_JOB_OK	The job ended successfully.
SPI_JOB_PENDING	The job is pending.
SPI_JOB_FAILED	The job has failed.

3.8.3.7 Function Spi_GetSequenceResult

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

Details:

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

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

Return: Spi_SeqResultType.

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

Prototype: `Spi_SeqResultType Spi_GetSequenceResult(Spi_SequenceType Sequence);`

Table 3-71. Spi_GetSequenceResult Arguments

Type	Name	Direction	Description
`Spi_SequenceType`	Sequence	**input**	Sequence ID.

Table 3-72. Spi_GetSequenceResult Return Values

Name	Description
SPI_SEQ_OK	The sequence ended successfully.
SPI_SEQ_PENDING	The sequence is pending.
SPI_SEQ_FAILED	The sequence has failed.

3.8.3.8 Function Spi_GetStatus

This function returns the status of the SPI driver.

Details:

This function returns the status of the SPI driver.

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

Return: Spi_StatusType.

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

Prototype: `Spi_StatusType Spi_GetStatus(void);`

Table 3-73. Spi_GetStatus Return Values

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

3.8.3.9 Function Spi_GetVersionInfo

This function returns the version information for the SPI driver.

Details:

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.

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

Table 3-74. Spi_GetVersionInfo Arguments

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

3.8.3.10 Function Spi_Init

This function initializes the SPI driver. It always require a valid pointer.

Details:

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

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

Prototype: `void Spi_Init(const Spi_ConfigType *ConfigPtr);`

Table 3-75. Spi_Init Arguments

Type	Name	Direction	Description
const`Spi_ConfigType`*	ConfigPtr	**input**	Specifies the pointer to the configuration set. It always require a valid pointer

3.8.3.11 Function Spi_JobTransferFinished

This function is called after a Job has been executed.

Details:

End Job transmission notification handler declaration.

The function calls Job and Sequence end notifications and schedules the next job of the sequence or on the liberated HW Unit.

The function (global) is implemented in `Spi.c` (Autosar Driver Layer).

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

Prototype: `void Spi_JobTransferFinished(const Spi_JobConfig *pJobConfig);`

Table 3-76. Spi_JobTransferFinished Arguments

Type	Name	Direction	Description
const`Spi_JobConfig`*	pJobConfig	**input**	The just transmited job pointer.
const`Spi_JobConfig`*	pJobConfig	**input**	The just transmited job pointer.

3.8.3.12 Function Spi_LockJobs

This function is called in order to mark the jobs of a sequence as ready to be transmitted.

Details:

For each job in sequence, the function checks if it is already linked to another pending sequence. If at least one job is already linked, the function returns E_NOT_OK. Elsewhere, all jobs in sequence are locked (linked to the current sequence)

Return: Std_ReturnType.

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

Prototype: `LOCAL_INLINE Std_ReturnType Spi_LockJobs(Spi_SequenceType Sequence, const Spi_SequenceConfig *pSequence);`

Table 3-77. Spi_LockJobs Arguments

Type	Name	Direction	Description
`Spi_SequenceType`	Sequence	**input**	The sequence ID.
const`Spi_SequenceConfig`*	pSequence	**input**	The sequence configuration.

Table 3-78. Spi_LockJobs Return Values

Name	Description
E_OK	The given sequence does not share its jobs with some other sequences, and all its jobs were successfully locked.
E_NOT_OK	The given sequence shares its jobs with some other sequences. No lock performed for its jobs.

3.8.3.13 Function Spi_MainFunction_Handling

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

Details:

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

- Service ID: 0x10

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

Prototype: `void Spi_MainFunction_Handling(void);`

3.8.3.14 Function Spi_ReadIB

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

Details:

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

Return: Std_ReturnType.

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

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

Table 3-79. Spi_ReadIB Arguments

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

Table 3-80. Spi_ReadIB Return Values

Name	Description
E_OK	Read command has been accepted.
E_NOT_OK	Read command has not been accepted.

3.8.3.15 Function Spi_ScheduleJob

This function will schedule a job for a given HW unit.

Details:

If the HWUnit is not busy, the transfer is started and the HW unit is marked as busy. If the HWUnit is not busy (another job is in progress), the new job is scheduled in a waiting job list, according to its priority.

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

Prototype: `LOCAL_INLINE void Spi_ScheduleJob(Spi_HWUnitQueue *pHWUnitQueue, Spi_JobType Job, const Spi_JobConfig *pJobConfig);`

Table 3-81. Spi_ScheduleJob Arguments

Type	Name	Direction	Description
`Spi_HWUnitQueue*`	pHWUnitQueue	**input**	HW Unit to be used by the job.
`Spi_JobType`	Job	**input**	ID of the scheduled job.

Table continues on the next page...

Table 3-81. Spi_ScheduleJob Arguments (continued)

Type	Name	Direction	Description
const`Spi_JobConfig`*	pJobConfig	**input**	Configuration of the scheduled job.

3.8.3.16 Function Spi_ScheduleNextJob

This function starts the transfer of the first scheduled job for a given HW unit.

Details:

If the list of scheduled jobs is not empty, pop the first job and start the transfer. Elsewhere, mark the HW unit as IDLE.

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

Prototype: `void Spi_ScheduleNextJob(Spi_HWUnitQueue *pHWUnitQueue);`

Table 3-82. Spi_ScheduleNextJob Arguments

Type	Name	Direction	Description
`Spi_HWUnitQueue`*	pHWUnitQueue	**input**	The HW Unit used for scheduling.

3.8.3.17 Function Spi_SetAsyncMode

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

Details:

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

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

Return: Std_ReturnType.

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

Prototype: `Std_ReturnType Spi_SetAsyncMode(Spi_AsyncModeType AsyncMode);`

Table 3-83. 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).

Table 3-84. Spi_SetAsyncMode Return Values

Name	Description
E_OK	The command ended successfully.
E_NOT_OK	The command has failed.

3.8.3.18 Function Spi_SetHWUnitAsyncMode

This function specifies the asynchronous mode for a given HWUnit.

Details:

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)

Return: Std_ReturnType.

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

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

Table 3-85. 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).

Table 3-86. Spi_SetHWUnitAsyncMode Return Values

Name	Description
E_OK	The command ended successfully.
E_NOT_OK	The command has failed.

3.8.3.19 Function Spi_SetupEB

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

Details:

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

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

Return: Std_ReturnType.

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

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

Table 3-87. Spi_SetupEB Arguments

Type	Name	Direction	Description
`Spi_ChannelType`	Channel	**input**	Channel ID.
const Spi_DataBufferType *	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.

Table continues on the next page...

Table 3-87. Spi_SetupEB Arguments (continued)

Type	Name	Direction	Description
Spi_DataBufferType *	DesDataBufferPtr	**output**	Pointer to the memory location that will hold the received data.

Table 3-88. Spi_SetupEB Return Values

Name	Description
E_OK	Setup command has been accepted.
E_NOT_OK	Setup command has not been accepted.

3.8.3.20 Function Spi_SyncTransmit

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

Details:

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

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

Return: Std_ReturnType.

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

Prototype: `Std_ReturnType Spi_SyncTransmit(Spi_SequenceType Sequence);`

Table 3-89. Spi_SyncTransmit Arguments

Type	Name	Direction	Description
`Spi_SequenceType`	Sequence	**input**	Sequence ID.

Table 3-90. Spi_SyncTransmit Return Values

Name	Description
E_OK	Transmission command has been completed successfully.
E_NOT_OK	Transmission command has not been accepted.

3.8.3.21 Function Spi_UnlockRemainingJobs

This function is called to release the jobs at the end of an async sequence transmission.

Details:

Mark the linked sequence for all jobs as NULL_PTR.

Pre: Pre-compile parameter SPI_CANCEL_API shall be STD_ON.

Prototype: `LOCAL_INLINE void Spi_UnlockRemainingJobs(Spi_JobType RemainingJobs, const Spi_SequenceConfig *pSequence);`

Table 3-91. Spi_UnlockRemainingJobs Arguments

Type	Name	Direction	Description
`Spi_JobType`	RemainingJobs	**input**	The starting job.
const`Spi_SequenceConfig`*	pSequence	**input**	The sequence configuration.

3.8.3.22 Function Spi_WriteIB

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

Details:

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

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

Return: Std_ReturnType.

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

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

Table 3-92. Spi_WriteIB Arguments

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

Table 3-93. Spi_WriteIB Return Values

Name	Description
E_OK	Command has been accepted.
E_NOT_OK	Command has not been accepted.

3.8.3.23 Function Spi_Dspi_IsrTCF_DSPI_0

This function is the Transfer Complete for DSPI 0. An interrupt will be generated at every frame transmitted.

Details:

Non-AutoSar support function used by interrupt service routine of the transfer complete Rx for DSPI 0

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter DSPI_0_ENABLED shall be STD_ON.

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter DSPI_0_ENABLED shall be STD_ON.

Prototype: `void Spi_Dspi_IsrTCF_DSPI_0(void);`

3.8.3.24 Function Spi_Dspi_IsrTCF_DSPI_1

This function is the Transfer Complete for DSPI 1. An interrupt will be generated at every frame transmitted..

Details:

Non-AutoSar support function used by interrupt service routine of the transfer complete Rx for DSPI 1

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter DSPI_1_ENABLED shall be STD_ON.

Prototype: `void Spi_Dspi_IsrTCF_DSPI_1(void);`

3.8.3.25 Function Spi_Dspi_IsrTCF_DSPI_2

This function is the Transfer Complete for DSPI 2. An interrupt will be generated at every frame transmitted.

Details:

Non-AutoSar support function used by interrupt service routine of the transfer complete Rx for DSPI 2.

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter DSPI_2_ENABLED shall be STD_ON.

Prototype: `void Spi_Dspi_IsrTCF_DSPI_2(void);`

3.8.3.26 Function Spi_Dspi_IsrTCF_DSPI_3

This function is the Transfer Complete for DSPI 3. An interrupt will be generated at every frame transmitted.

Details:

Non-AutoSar support function used by interrupt service routine of the transfer complete Rx for DSPI 3.

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter DSPI_3_ENABLED shall be STD_ON.

Prototype: `void Spi_Dspi_IsrTCF_DSPI_3(void);`

3.8.3.27 Function Spi_Dspi_IsrTCF_SPI_0

This function is the Transfer Complete for SPI 0. An interrupt will be generated at every frame transmitted.

Details:

Non-AutoSar support function used by interrupt service routine of the transfer complete Rx for SPI 0

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter SPI_0_ENABLED shall be STD_ON.

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter SPI_0_ENABLED shall be STD_ON.

Prototype: `void Spi_Dspi_IsrTCF_SPI_0(void);`

3.8.3.28 Function Spi_Dspi_IsrTCF_SPI_1

This function is the Transfer Complete for SPI 1. An interrupt will be generated at every frame transmitted..

Details:

Non-AutoSar support function used by interrupt service routine of the transfer complete Rx for SPI 1

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter SPI_1_ENABLED shall be STD_ON.

Prototype: `void Spi_Dspi_IsrTCF_SPI_1(void);`

3.8.3.29 Function Spi_Dspi_IsrTCF_SPI_2

This function is the Transfer Complete for SPI 2. An interrupt will be generated at every frame transmitted.

Details:

Non-AutoSar support function used by interrupt service routine of the transfer complete Rx for SPI 2.

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter SPI_2_ENABLED shall be STD_ON.

Prototype: `void Spi_Dspi_IsrTCF_SPI_2(void);`

3.8.3.30 Function Spi_Dspi_IsrTCF_SPI_3

This function is the Transfer Complete for SPI 3. An interrupt will be generated at every frame transmitted.

Details:

Non-AutoSar support function used by interrupt service routine of the transfer complete Rx for SPI 3.

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter SPI_3_ENABLED shall be STD_ON.

Prototype: `void Spi_Dspi_IsrTCF_SPI_3(void);`

3.8.3.31 Function Spi_Dspi_IsrTCF_SPI_4

This function is the Transfer Complete for SPI 4. An interrupt will be generated at every frame transmitted.

Details:

Non-AutoSar support function used by interrupt service routine of the transfer complete Rx for SPI 4

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter SPI_4_ENABLED shall be STD_ON.

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter SPI_4_ENABLED shall be STD_ON.

Prototype: `void Spi_Dspi_IsrTCF_SPI_4(void);`

3.8.3.32 Function Spi_Dspi_IsrTCF_SPI_5

This function is the Transfer Complete for SPI 5. An interrupt will be generated at every frame transmitted.

Details:

Non-AutoSar support function used by interrupt service routine of the transfer complete Rx for SPI 5

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter SPI_5_ENABLED shall be STD_ON.

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter SPI_5_ENABLED shall be STD_ON.

Prototype: `void Spi_Dspi_IsrTCF_SPI_5(void);`

3.8.3.33 Function Spi_Dspi_IsrRxDma_DSPI_0

This function is the DMA Rx notification for the DSPI 0.

Details:

Non-AutoSar support function used by MCL interrupt serive routine for the DMA Rx for DSPI 0

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter DSPI_0_ENABLED shall be STD_ON.

Prototype: `void Spi_Dspi_IsrRxDma_DSPI_0(void);`

3.8.3.34 Function Spi_Dspi_IsrRxDma_DSPI_1

This function is the DMA Rx notification for the DSPI 1.

Details:

Non-AutoSar support function used by MCL interrupt serive routine for the DMA Rx for DSPI 1

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter DSPI_1_ENABLED shall be STD_ON.

Prototype: `void Spi_Dspi_IsrRxDma_DSPI_1(void);`

3.8.3.35 Function Spi_Dspi_IsrRxDma_DSPI_2

This function is the DMA Rx notification for the DSPI 2.

Details:

Non-AutoSar support function used by MCL interrupt serive routine for the DMA Rx for DSPI 2

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter DSPI_2_ENABLED shall be STD_ON.

Prototype: `void Spi_Dspi_IsrRxDma_DSPI_2(void);`

3.8.3.36 Function Spi_Dspi_IsrRxDma_DSPI_3

This function is the DMA Rx notification for the DSPI 3.

Details:

Non-AutoSar support function used by MCL interrupt serive routine for the DMA Rx for DSPI 3

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter DSPI_3_ENABLED shall be STD_ON.

Prototype: `void Spi_Dspi_IsrRxDma_DSPI_3(void);`

3.8.3.37 Function Spi_Dspi_IsrRxDma_SPI_0

This function is the DMA Rx notification for the SPI 0.

Details:

Non-AutoSar support function used by MCL interrupt serive routine for the DMA Rx for SPI 0

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter SPI_0_ENABLED shall be STD_ON.

Prototype: `void Spi_Dspi_IsrRxDma_SPI_0(void);`

3.8.3.38 Function Spi_Dspi_IsrRxDma_SPI_1

This function is the DMA Rx notification for the SPI 1.

Details:

Non-AutoSar support function used by MCL interrupt serive routine for the DMA Rx for SPI 1

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter SPI_1_ENABLED shall be STD_ON.

Prototype: `void Spi_Dspi_IsrRxDma_SPI_1(void);`

3.8.3.39 Function Spi_Dspi_IsrRxDma_SPI_2

This function is the DMA Rx notification for the SPI 2.

Details:

Non-AutoSar support function used by MCL interrupt serive routine for the DMA Rx for SPI 2

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter SPI_2_ENABLED shall be STD_ON.

Prototype: `void Spi_Dspi_IsrRxDma_SPI_2(void);`

3.8.3.40 Function Spi_Dspi_IsrRxDma_SPI_3

This function is the DMA Rx notification for the SPI 3.

Details:

Non-AutoSar support function used by MCL interrupt serive routine for the DMA Rx for SPI 3

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter SPI_3_ENABLED shall be STD_ON.

Prototype: `void Spi_Dspi_IsrRxDma_SPI_3(void);`

3.8.3.41 Function Spi_Dspi_IsrRxDma_SPI_4

This function is the DMA Rx notification for the SPI 4.

Details:

Non-AutoSar support function used by MCL interrupt serive routine for the DMA Rx for SPI 4

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter SPI_4_ENABLED shall be STD_ON.

Prototype: `void Spi_Dspi_IsrRxDma_SPI_4(void);`

3.8.3.42 Function Spi_Dspi_IsrRxDma_SPI_5

This function is the DMA Rx notification for the SPI 5.

Details:

Non-AutoSar support function used by MCL interrupt serive routine for the DMA Rx for SPI 5

Pre: Pre-compile parameter SPI_LEVEL_DELIVERED shall be LEVEL1 or LEVEL2. Pre-compile parameter SPI_5_ENABLED shall be STD_ON.

Prototype: `void Spi_Dspi_IsrRxDma_SPI_5(void);`

3.8.4 Structs Reference

Data structures supported by the driver are as per AUTOSAR SPI Driver software specification Version 4.2 Rev0002 .

3.8.4.1 Structure Spi_BufferDescriptorType

The structure contains the pointers to the Tx/Rx memory locations for the given buffer (IB or EB).

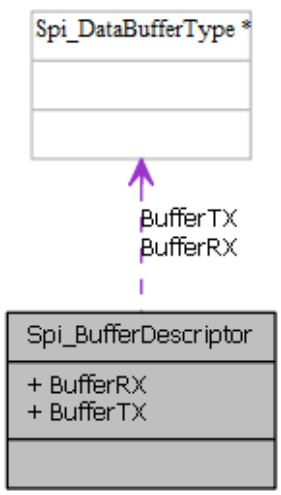

Figure 3-1. Struct Spi_BufferDescriptorType

Declaration

```
typedef struct
{
  Spi_DataBufferType * pBufferRX,
  Spi_DataBufferType * pBufferTX
} Spi_BufferDescriptorType;
```

Table 3-94. Structure Spi_BufferDescriptorType member description

Member	Description
pBufferRX	Receive buffer pointer.
pBufferTX	Transmit buffer pointer.

3.8.4.2 Structure Spi_ChannelConfigType

The structure contains the channel configuration parameters.

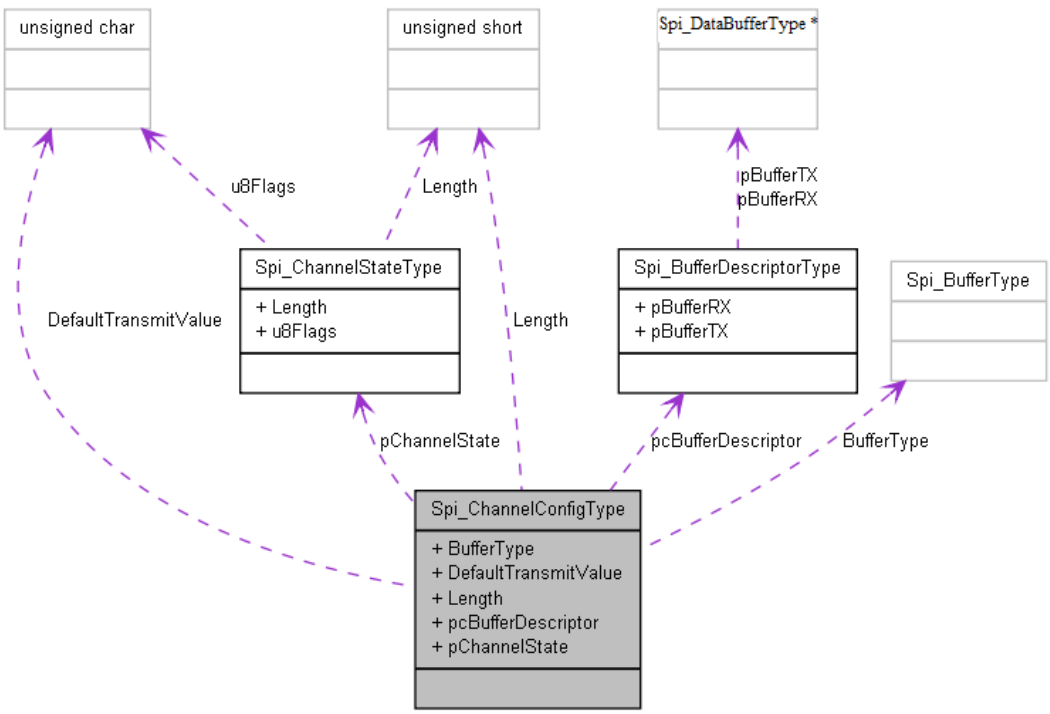

Figure 3-2. Struct Spi_ChannelConfigType

Declaration

```
typedef struct
{
  Spi_BufferType BufferType,
  Spi_DataBufferType DefaultTransmitValue,
  Spi_NumberOfDataType Length,
  Spi_BufferDescriptorType * pcBufferDescriptor,
  Spi_ChannelStateType * pChannelState
  Spi_NumberOfDataType FrameCnt
} Spi_ChannelConfigType;
```

Table 3-95. Structure Spi_ChannelConfigType member description

Member	Description
BufferType	Buffer Type IB/EB.
DefaultTransmitValue	Default Transmit Value.
Length	Data length.
pcBufferDescriptor	Buffer Descriptor.
pChannelState	Implementation specific field referencing the channel internal state.
FrameCnt	Total frame can be stored in EB buffer. This field is uses when enable high speed support for slave mode.

3.8.4.3 Structure Spi_ChannelStateType

Internal structure used to manage the channel state.

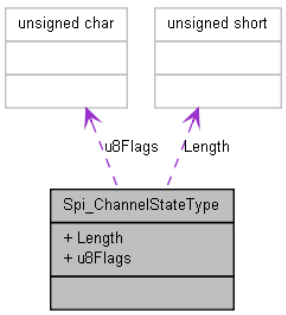

Figure 3-3. Struct Spi_ChannelStateType

Declaration

```
typedef struct
{
  Spi_NumberOfDataType Length,
  uint8 u8Flags
} Spi_ChannelStateType;
```

Table 3-96. Structure Spi_ChannelStateType member description

Member	Description
Length	Actual Transfer size for EB.
u8Flags	Default Transmit Enabled.

3.8.4.4 Structure Spi_ConfigType

This is the top level structure containing all the needed parameters for the SPI Handler Driver.

This is the top level structure containing all the needed parameters for the SPI Handler/ Driver.

Declaration:

```
typedef struct
{
  const Spi_AttributesConfigType * pcAttributesConfig,
  const Spi_ChannelConfigType (* const pcChannelConfig)[],
  const Spi_HWUnitConfigType (* const pcHWUnitConfig)[],
  const Spi_JobConfigType (* const pcJobConfig)[],
  const Spi_SequenceConfigType (* const pcSequenceConfig)[],
  const Mcal_DemErrorType Spi_E_Hardware_ErrorCfg,
```

```
  Spi_ChannelType Spi_Max_Channel,
  Spi_JobType Spi_Max_Job,
  Spi_SequenceType Spi_Max_Sequence,
  uint16 u16MaxExternalDevice,
  const Spi_TSBConfigType * pcJobTSBConfig
} Spi_ConfigType;
```

Table 3-97. Structure Spi_ConfigType member description

Member	Description
pcAttributesConfig	Channel & SPI HW unit attributes.
pcChannelConfig	Array of channels defined in the configuration.
pcHWUnitConfig	Array of LLD SPI device instances.
pcJobConfig	Array of jobs defined in the configuration.
pcSequenceConfig	Array of sequences defined in the configuration.
Spi_E_Hardware_ErrorCfg	SPI Driver DEM Error: SPI_E_HARDWARE_ERROR.
Spi_Max_Channel	Number of channels defined in the configuration.
Spi_Max_Job	Number of jobs defined in the configuration.
Spi_Max_Sequence	Number of sequences defined in the configuration.
u16MaxExternalDevice	Number of external devices defined in the configuration.
pcJobTSBConfig	Array of TSBJobs defined in the configuration.

3.8.4.5 Structure Spi_HWUnitConfigType

This structure holds the HWUnit configuration parameters.

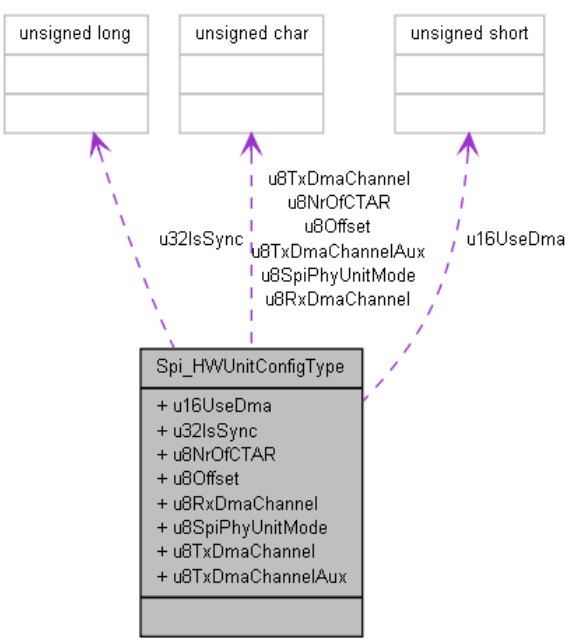

Figure 3-4. Struct Spi_HWUnitConfigType

Declaration

```
typedef struct
{
  uint16 u16UseDma,
  uint32 u32IsSync,
  uint8 u8NrOfCTAR,
  uint8 u8Offset,
  uint8 u8RxDmaChannel,
  uint8 u8SpiPhyUnitMode,
  uint8 u8TxDmaChannel,
  uint8 u8TxDmaChannelAux
} Spi_HWUnitConfigType;
```

Table 3-98. Structure Spi_HWUnitConfigType member description

Member	Description
u16UseDma	Flag indicating if DMA will be used or not for this SPI unit.
u32IsSync	Indicates if the HW unit is configured as Sync or Async.
u8NrOfCTAR	Indicates the number of CTAR registers available for the SPI module.
u8Offset	SPI HWunit physical offset on SOC.
u8RxDmaChannel	RX DMA channel - enabled by the SPI RX source.
u8SpiPhyUnitMode	Slave Mode - enabled.
u8TxDmaChannel	Master TX DMA channel - enabled by the SPI TX source.
u8TxDmaChannelAux	Auxiliary TX DMA channel - triggered by the master TX Dma.

3.8.4.6 Structure Spi_HWUnitQueue

This structure holds the HWUnit scheduling queue.

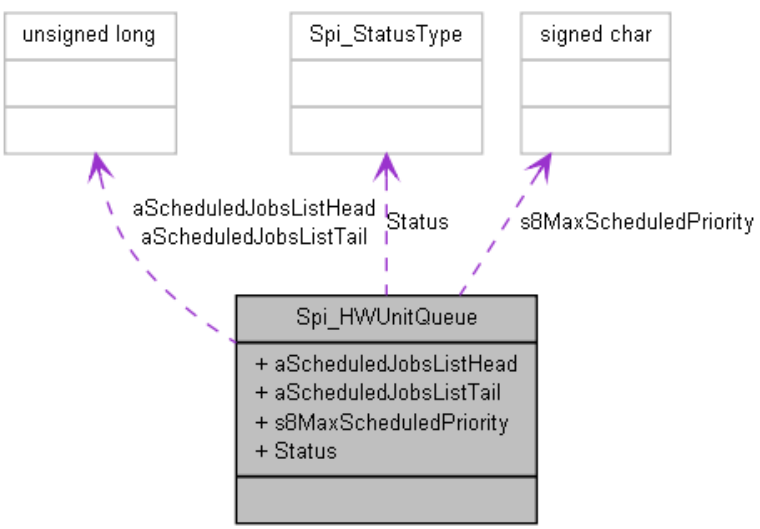

Figure 3-5. Struct Spi_HWUnitQueue

Details:

For async transmissions, this structure holds the HWUnit scheduling queue . For sync transmissions, only HWUnit Status is managed.

Declaration:

```
typedef struct
{
  Spi_JobType aScheduledJobsListHead[SPI_JOB_PRIORITY_LEVELS_COUNT],
  Spi_JobType aScheduledJobsListTail[SPI_JOB_PRIORITY_LEVELS_COUNT],
  sint8 s8MaxScheduledPriority,
  Spi_StatusType Status
} Spi_HWUnitQueue;
```

Table 3-99. Structure Spi_HWUnitQueue member description

Member	Description
aScheduledJobsListHead	Array of the IDs of jobs to be scheduled, for each priority level.
aScheduledJobsListTail	Array of the IDs of last jobs in queues, for each priority level.
s8MaxScheduledPriority	Array of the IDs of last jobs in queues, for each priority level.
Status	SPI state.

3.8.4.7 Structure Spi_JobConfigType

This is the structure containing all the parameters needed to completely define a Job.

Declaration

```
typedef struct
{
  Spi_NotifyType * pfEndNotification,
  Spi_ExternalDeviceType ExternalDevice,
  Spi_Ipw_DeviceAttributesConfigType ExternalDeviceAttrs,
  Spi_HWUnitType HWUnit,
  Spi_ChannelType NumChannels,
  const Spi_ChannelType (* const pcChannelIndexList)[],
  Spi_JobStateType * pJobState,
  sint8 s8Priority,
  Spi_NotifyType * pfStartNotification,
  uint32 u32HWoffset
} Spi_JobConfigType;
```

Table 3-100. Structure Spi_JobConfigType member description

Member	Description
pfEndNotification	Job end notification.
ExternalDevice	ExternalDevice
ExternalDeviceAttrs	Implementation specific field: cached LLD device attributes.
HWUnit	HWUnit.
NumChannels	Number of channels in the job.
pcChannelIndexList	Channel index list.
pJobState	Implementation specific field referencing the channel internal state

Table continues on the next page...

Table 3-100. Structure Spi_JobConfigType member description (continued)

Member	Description
s8Priority	Priority.
pfStartNotification	Job start notification.
u32HWoffset	HW Unit offset.

3.8.4.8 Structure Spi_JobStateType

Internal structure used to manage the job state.

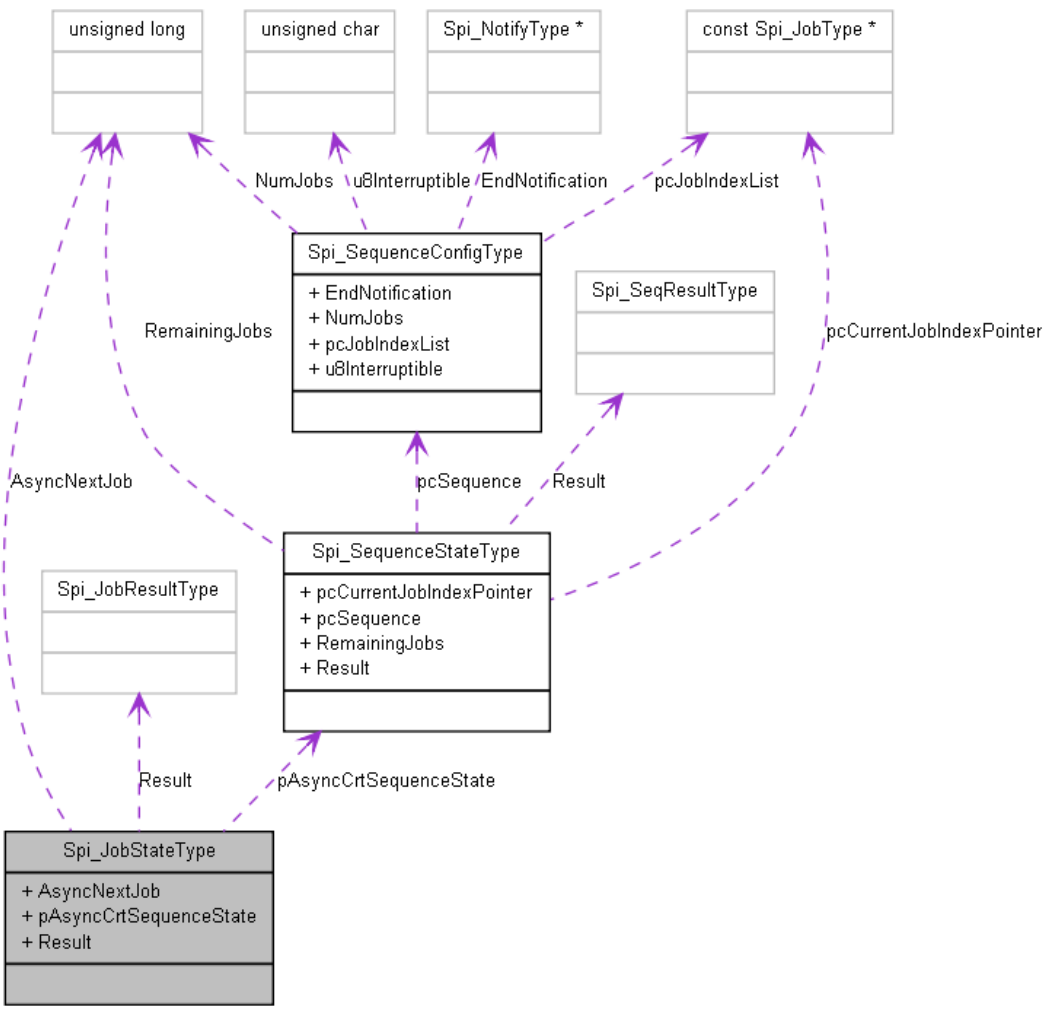

Figure 3-6. Struct Spi_JobStateType

Declaration

```
typedef struct
{
  Spi_JobType AsyncNextJob,
  Spi_SequenceStateType * pAsyncCrtSequenceState,
```

```
  Spi_JobResultType Result
} Spi_JobStateType;
```

Table 3-101. Structure Spi_JobStateType member description

Member	Description
AsyncNextJob	Pointer to the next async job planned for transmission.
pAsyncCrtSequenceState	Pointer to the state information of the async sequence
Result	Job Result.

3.8.4.9 Structure Spi_SequenceConfigType

This structure contains all the needed data to configure one SPI Sequence.

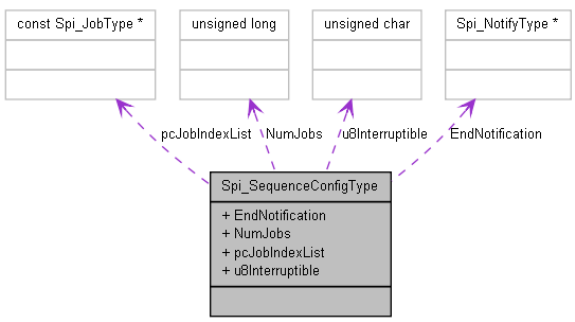

Figure 3-7. Struct Spi_SequenceConfigType

Declaration

```
typedef struct
{
  Spi_NotifyType * pfEndNotification,
  Spi_JobType NumJobs,
  const Spi_JobType (* const pcJobIndexList)[],
  uint8 u8Interruptible,
  Spi_HPNotifyType * pfHPEndNotification
} Spi_SequenceConfigType;
```

Table 3-102. Structure Spi_SequenceConfigType member description

Member	Description
pfEndNotification	Job notification handler.
NumJobs	Number of jobs in the sequence.
pcJobIndexList	ob index list.
u8Interruptible	Boolean indicating if the Sequence is interruptible or not.
pfHPEndNotification	Job notification handler support for high speed in slave mode.

3.8.4.10 Structure Spi_SequenceStateType

Internal structure used to manage the sequence state.

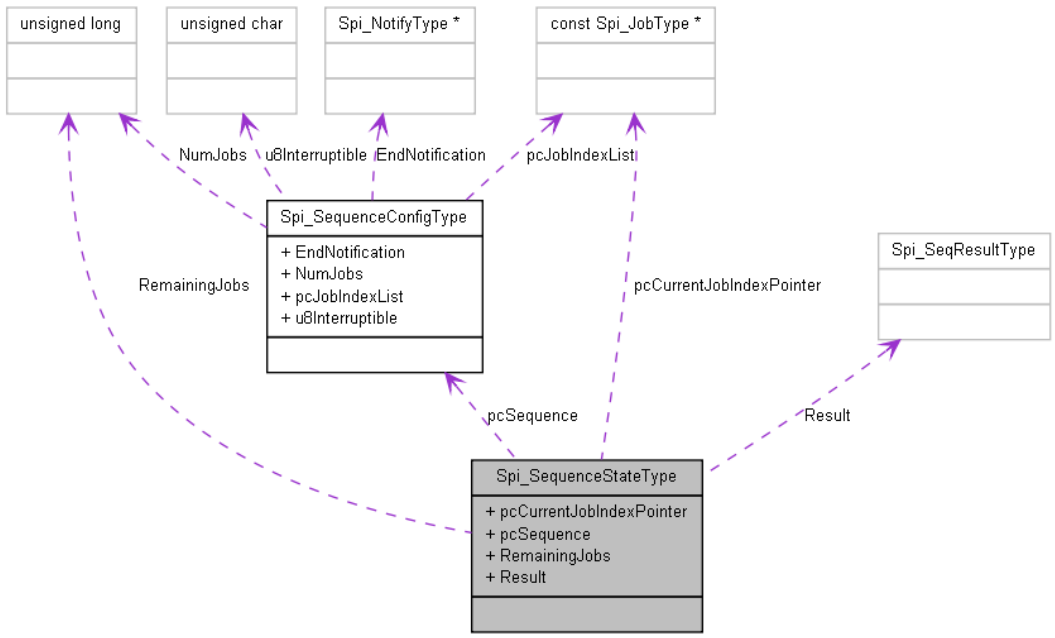

Figure 3-8. Struct Spi_SequenceState

Declaration

```
typedef struct
{
  const Spi_JobType * pcCurrentJobIndexPointer,
  const Spi_SequenceConfigType * pcSequence,
  Spi_JobType RemainingJobs,
  Spi_SeqResultType Result
} Spi_SequenceStateType;
```

Table 3-103. Structure Spi_SequenceStateType member description

Member	Description
pcCurrentJobIndexPointer	Position in JobIndexList to the job in transmission of an async sequence.
pcSequence	Pointer to the configuration.
RemainingJobs	Number of jobs in a pending async sequence, not yet transmitted.
Result	Seq Result.

3.8.5 Types Reference

Types supported by the driver are as per AUTOSAR SPI Driver software specification Version 4.2 Rev0002 .

3.8.5.1 Typedef Spi_ChannelType

Specifies the identification (ID) for a Channel.

Type: uint8

3.8.5.2 Typedef Spi_DataBufferType

Type: uint8

3.8.5.3 Typedef Spi_ExternalDeviceType

Contains the ID of an external device.

Details:

This contains the identification (ID) of the external device for which there's a collection of particular settings

Type: uint8

3.8.5.4 Typedef Spi_HWUnitType

Specifies the ID for a SPI Hardware microcontroller peripheral unit.

Details:

This type is used for specifying the identification (ID) for a SPI Hardware microcontroller peripheral unit.

Type: uint8

3.8.5.5 Typedef Spi_JobType

Specifies the identification (ID) for a Job.

Type: uint16

3.8.5.6 Typedef Spi_NotifyType

Type: void(

3.8.5.7 Typedef Spi_NumberOfDataType

Type for defining the number of data elements of the type Spi_DataBufferType.

Details:

Type for defining the number of data elements of the type Spi_DataBufferType to send and/or receive by Channel.

Type: uint16

3.8.5.8 Typedef Spi_SequenceType

Specifies the identification (ID) for a sequence of jobs.

Type: uint8

3.8.6 Variables Reference

Variables supported by the driver are as per AUTOSAR SPI Driver software specification Version 4.2 Rev0002 .

3.8.6.1 Variable Spi_u32SpiBusySyncHWUnitsStatus

Spi Sync Transmit Running HWUnits Status.

Declaration:

```
static volatile uint32 Spi_u32SpiBusySyncHWUnitsStatus
```

3.8.6.2 Variable Spi_aSpiChannelState

Declaration:

```
Spi_ChannelStateType Spi_aSpiChannelState[SPI_MAX_CHANNEL]
```

3.8.6.3 Variable Spi_pcSpiConfigPtr

Pointer initialized during init with the address of the received configuration structure.

Details:

Will be used by all functions to access the configuration data.

Declaration:

```
const Spi_ConfigType* Spi_pcSpiConfigPtr
```

3.8.6.4 Variable Spi_aSpiHWUnitQueueArray

Array of HW units queues.

Declaration:

```
Spi_HWUnitQueue Spi_aSpiHWUnitQueueArray[SPI_MAX_HWUNIT]
```

3.8.6.5 Variable Spi_aSpiJobState

Spi State.

Declaration:

```
Spi_JobStateType Spi_aSpiJobState[SPI_MAX_JOB]
```

3.8.6.6 Variable Spi_aSpiSequenceState

Spi State.

Declaration:

```
Spi_SequenceStateType Spi_aSpiSequenceState[SPI_MAX_SEQUENCE]
```

3.8.6.7 Variable Spi_au32SpiSeqUsedHWUnits

Note

Array of used HW units per sequence: The element corresponding to a given sequence will have asserted the bits corresponding to the used HW units.

Declaration:

```
uint32 Spi_au32SpiSeqUsedHWUnits[SPI_MAX_SEQUENCE]
```

3.8.6.8 Variable Spi_aSpiJobTSBConfig

Job Configuration of TSB mode.

Declaration:

```
const Spi_TSBConfigType Spi_aSpiJobTSBConfig[TSBJobCount]
```

3.9 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).

Chapter 4
Tresos Configuration Plug-in

This chapter describes the Tresos configuration plug-in for the SPI Driver. The most of the parameters are described below.

4.1 Configuration elements of Spi

Included forms :

- IMPLEMENTATION_CONFIG_VARIANT
- SpiPublishedInformation
- SpiGeneral
- SpiDemEventParameterRefs
- SpiNonAUTOSAR
- SpiDriver

Table 4-1. Revision table

Revision	Date
0.8.1	

4.2 Form IMPLEMENTATION_CONFIG_VARIANT

VariantPreCompile: Only parameters with "Pre-compile time" configuration are allowed in this variant. VariantPostBuild: Parameters with "Pre-compile time", "Link time" and "Post-build time" are allowed in this variant. VariantLinkTime: Only parameters with "Pre-compile time" and "Link time" are allowed in this variant.

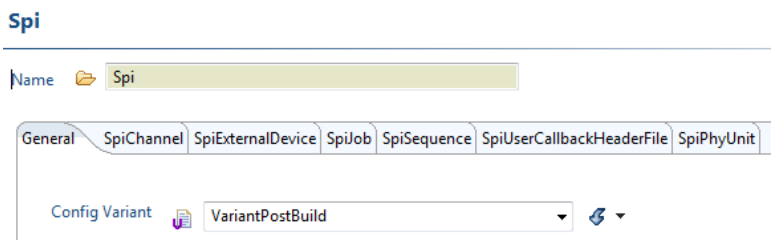

Figure 4-1. Tresos Plugin snapshot for IMPLEMENTATION_CONFIG_VARIANT form.

Table 4-2. Attribute IMPLEMENTATION_CONFIG_VARIANT detailed description

Property	Value
Label	Config Variant
Type	ENUMERATION
Symbolic Name	false
Default	VariantPreCompile
Range	VariantPreCompile VariantPostBuild VariantLinkTime

4.3 Form CommonPublishedInformation

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

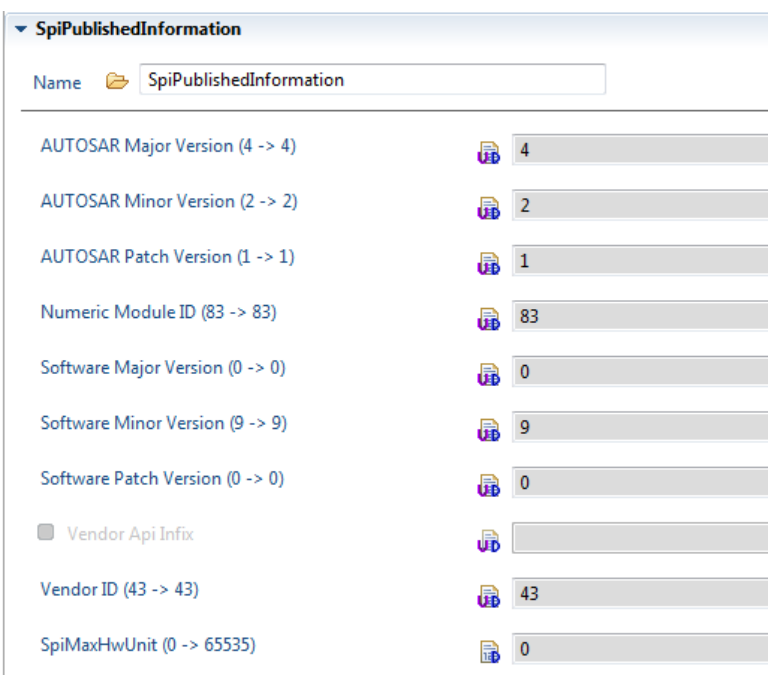

Figure 4-2. Tresos Plugin snapshot for CommonPublishedInformation form.

4.3.1 ArReleaseMajorVersion (CommonPublishedInformation)

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

Table 4-3. Attribute ArReleaseMajorVersion (CommonPublishedInformation) detailed description

Property	Value
Label	AUTOSAR Major Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	4
Invalid	Range >=4 <=4

4.3.2 ArReleaseMinorVersion (CommonPublishedInformation)

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

Table 4-4. Attribute ArReleaseMinorVersion (CommonPublishedInformation) detailed description

Property	Value
Label	AUTOSAR Minor Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	2
Invalid	Range >=2 <=2

4.3.3 ArReleaseRevisionVersion (CommonPublishedInformation)

Revision version number of AUTOSAR specification on which the appropriate implementation is based on.

Table 4-5. Attribute ArReleaseRevisionVersion (CommonPublishedInformation) detailed description

Property	Value
Label	AUTOSAR Release Revision Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	2
Invalid	Range >=2 <=2

4.3.4 ModuleId (CommonPublishedInformation)

Module ID of this module from Module List.

Table 4-6. Attribute ModuleId (CommonPublishedInformation) detailed description

Property	Value
Label	Module Id
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	83
Invalid	Range >=83 <=83

4.3.5 SwMajorVersion (CommonPublishedInformation)

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

Table 4-7. Attribute SwMajorVersion (CommonPublishedInformation) detailed description

Property	Value
Label	Software Major Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false

Table continues on the next page...

Table 4-7. Attribute SwMajorVersion (CommonPublishedInformation) detailed description (continued)

Property	Value
Default	1
Invalid	Range >=1 <=1

4.3.6 SwMinorVersion (CommonPublishedInformation)

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

Table 4-8. Attribute SwMinorVersion (CommonPublishedInformation) detailed description

Property	Value
Label	Software Minor Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	0
Invalid	Range >=0 <=0

4.3.7 SwPatchVersion (CommonPublishedInformation)

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

Table 4-9. Attribute SwPatchVersion (CommonPublishedInformation) detailed description

Property	Value
Label	Software Patch Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	0
Invalid	Range >=0 <=0

4.3.8 VendorApiInfix (CommonPublishedInformation)

In driver modules which can be instantiated several times on a single ECU, BSW00347 requires that the name of APIs is extended by the VendorId and a vendor specific name. This parameter is used to specify the vendor specific name. In total, the implementation specific name is generated as follows: <ModuleName>_>VendorId>_<VendorApiInfix><Api name from SWS>. E.g. assuming that the VendorId of the implementor is 123 and the implementer chose a VendorApiInfix of "v11r456" a api name Can_Write defined in the SWS will translate to Can_123_v11r456Write. This parameter is mandatory for all modules with upper multiplicity > 1. It shall not be used for modules with upper multiplicity =1.

Table 4-10. Attribute VendorApiInfix (CommonPublishedInformation) detailed description

Property	Value
Label	Vendor Api Infix
Type	STRING_LABEL
Origin	Custom
Symbolic Name	false
Default	
Enable	false

4.3.9 VendorId (CommonPublishedInformation)

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

Table 4-11. Attribute VendorId (CommonPublishedInformation) detailed description

Property	Value
Label	Vendor Id
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	43
Invalid	Range >=43 <=43

4.4 Form SpiGeneral

General configuration settings for SPI-Handler.

Included forms :

- Form SpiPhyUnit

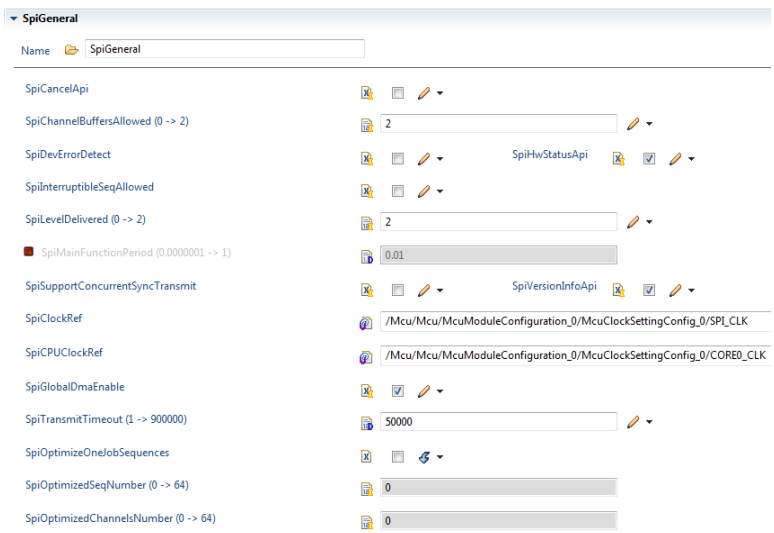

Figure 4-3. Tresos Plugin snapshot for SpiGeneral form.

4.4.1 SpiCancelApi (SpiGeneral)

Switches the Spi_Cancel function ON or OFF.

Table 4-12. Attribute SpiCancelApi (SpiGeneral) detailed description

Property	Value
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

4.4.2 SpiChannelBuffersAllowed (SpiGeneral)

Note

Selects the SPI Handler or Driver Channel Buffers usage allowed and delivered.

0 - Only Internal Buffers (IB) are allowed 1 - Only External buffers (EB) are allowed 2 - Both Internal (IB) and External (EB) buffers are allowed

Table 4-13. Attribute SpiChannelBuffersAllowed (SpiGeneral) detailed description

Property	Value
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	0
Invalid	Range <=2 >=0

4.4.3 SpiDevErrorDetect (SpiGeneral)

Switches the Development Error Detection and Notification ON or OFF.

Table 4-14. Attribute SpiDevErrorDetect (SpiGeneral) detailed description

Property	Value
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

4.4.4 SpiHwStatusApi (SpiGeneral)

Switches the Spi_GetHWUnitStatus function ON or OFF.

Table 4-15. Attribute SpiHwStatusApi (SpiGeneral) detailed description

Property	Value
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

4.4.5 SpiInterruptibleSeqAllowed (SpiGeneral)

Switches the Interruptible Sequences handling functionality ON or OFF.

Table 4-16. Attribute SpiInterruptibleSeqAllowed (SpiGeneral) detailed description

Property	Value
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.4.6 SpiLevelDelivered (SpiGeneral)

Note

Selects the SPI Handler or Driver level of scalable functionality that is available and delivered.

Level 0 Only Simple Synchronous Behavior Level 1 Basic Asynchronous Behaviour Level 2 Enhanced Behaviour

Table 4-17. Attribute SpiLevelDelivered (SpiGeneral) detailed description

Property	Value
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	2
Invalid	Range <=2 >=0

4.4.7 SpiSupportConcurrentSyncTransmit (SpiGeneral)

Specifies whether concurrent Spi_SyncTransmit() calls for different se-quences shall be configurable.

Table 4-18. Attribute SpiSupportConcurrentSyncTransmit (SpiGeneral) detailed description

Property	Value
Type	BOOLEAN

Table continues on the next page...

Table 4-18. Attribute SpiSupportConcurrentSyncTransmit (SpiGeneral) detailed description (continued)

Property	Value
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.4.8 SpiVersionInfoApi (SpiGeneral)

Switches the Spi_GetVersionInfo function ON or OFF.

Table 4-19. Attribute SpiVersionInfoApi (SpiGeneral) detailed description

Property	Value
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

4.4.9 SpiGlobalDmaEnable (SpiGeneral)

Note

If checked, it allows using the DMA module during the transfer. For each DSPI unit a transfering method can be configured: FIFO or DMA.

If not checked, all DSPI units will use FIFO transfering mode.

This is an implementation parameter.

Table 4-20. Attribute SpiGlobalDmaEnable (SpiGeneral) detailed description

Property	Value
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.4.10 SpiTransmitTimeout (SpiGeneral)

Note

Timeout value (microseconds) used by Spi_SyncTransmit and Spi_AsyncTransmit functions to wait for TX/RX transmission to complete one frame.

This is an implementation parameter. The transmission will be unsuccessful if the Chip cannot completly transfer one frame during this timeout. The precision of this value is quite low, it must be greater than the time needed to completly transmit one frame(TimePerFrame). Calculate TimePerFrame (microseconds) by formula below.

$$\text{TimePerFrame(us)} = (\text{SpiTimeCs2Clk} * 1000000) + (\frac{1000000}{\text{SpiBaudrate}} * \text{SpiDataWidth}) + (\text{SpiTimeClk2Cs} * 1000000)$$

Figure 4-4. If keep chip select asserted between frame transfers(SpiCsContinous = TRUE).

$$\text{TimePerFrame(us)} = (\text{SpiTimeCs2Clk} * 1000000) + (\frac{1000000}{\text{SpiBaudrate}} * \text{SpiDataWidth}) + (\text{SpiTimeClk2Cs} * 1000000) + (\text{TimeCs2Cs} * 1000000)$$

Figure 4-5. If don't keep chip select asserted between frame transfers(SpiCsContinous = FALSE).

Table 4-21. Attribute SpiTransmitTimeout (SpiGeneral) detailed description

Property	Value
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	50000
Invalid	Range <=900000 >=1

4.4.11 SpiOptimizeOneJobSequences (SpiGeneral)

Note

Activates the SPI transmission optimization for the sequences having only one job. At his 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.

This is an implementation parameter.

Table 4-22. Attribute SpiOptimizeOneJobSequences (SpiGeneral) detailed description

Property	Value
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.4.12 SpiOptimizedSeqNumber (SpiGeneral)

Note

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.

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

Table 4-23. Attribute SpiOptimizedSeqNumber (SpiGeneral) detailed description

Property	Value
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	0
Invalid	Range <=64 >=0

4.4.13 SpiOptimizedChannelsNumber (SpiGeneral)

Note

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.

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

Table 4-24. Attribute SpiOptimizedChannelsNumber (SpiGeneral) detailed description

Property	Value
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	0
Invalid	Range <=64 >=0

4.4.14 SpiClockRef (SpiGeneral)

Reference to the SPI clock source configuration, which is set into the MCU driver configuration. This clock source is used for configure SPI baudrate.

Table 4-25. Attribute SpiClockRef (SpiGeneral) detailed description

Property	Value
Type	REFERENCE
Origin	Custom

4.4.15 SpiCPUClockRef (SpiGeneral)

Reference to the CPU clock source configuration, which is set into the MCU driver configuration. This CPU clock source is used for configure Timeout value (in microseconds) in SpiTransmitTimeout used to wait for TX/RX transmission to complete one frame.

Table 4-26. Attribute SpiCPUClockRef (SpiGeneral) detailed description

Property	Value
Type	REFERENCE
Origin	Custom

4.4.16 SpiMainFunctionPeriod (SpiGeneral)

This parameter defines the cycle time of the function Spi_MainFunction_Handling in seconds. The parameter is not used by the driver it self, but it is used by upper layer.

Table 4-27. Attribute SpiMainFunctionPeriod (SpiGeneral) detailed description

Property	Value
Type	FLOAT
Origin	Custom
Symbolic Name	false
Default	0.01

4.4.17 Form SpiPhyUnit

Note

Logical to Physical SPI Bus mapping.

This is an implementation specific container.

Is included by form : Form SpiGeneral

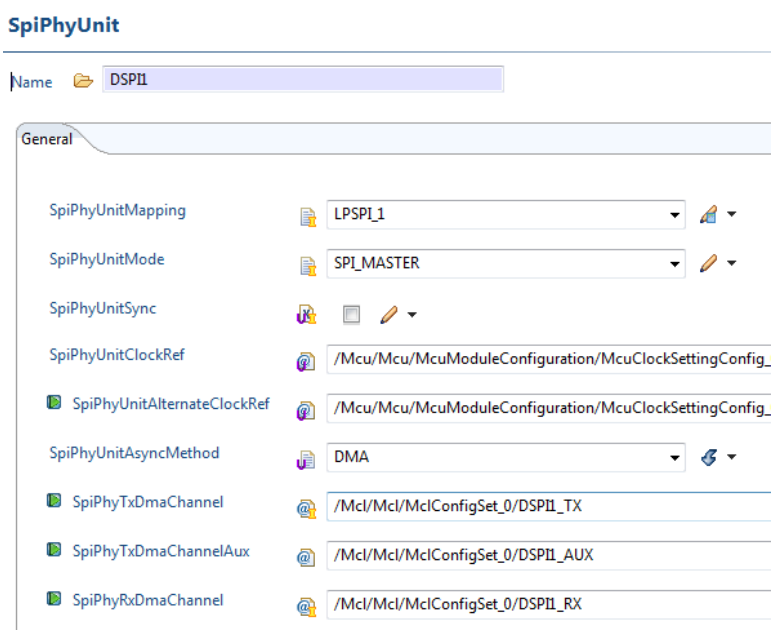

Figure 4-6. Tresos Plugin snapshot for SpiPhyUnit form.

4.4.17.1 SpiPhyUnitMapping (SpiPhyUnit)

Note

Logical SpiHWunit to physical DSPI_[0|1|2|3|4] assignment. It depends on the number of units present in the chip version.

This is an implementation specific parameter.

Table 4-28. Attribute SpiPhyUnitMapping (SpiPhyUnit) detailed description

Property	Value
Type	ENUMERATION
Origin	Custom
Symbolic Name	false

4.4.17.2 SpiPhyUnitMode (SpiPhyUnit)

Note

Select between SPI_MASTER and SPI_SLAVE modes.

This is an implementation specific parameter.

Table 4-29. Attribute SpiPhyUnitMode (SpiPhyUnit) detailed description

Property	Value
Type	ENUMERATION
Origin	Custom
Symbolic Name	false
Default	SPI_MASTER
Range	SPI_MASTER SPI_SLAVE

4.4.17.3 SpiPhyUnitSync (SpiPhyUnit)

Note

Specific if this HwUnit can only do sync transfers. If true then this hardware unit is dedicated for Synchronous transfers only. If false then this hardware unit is dedicated for Asynchronous transfers only. False is applicable only if SpiGeneral/SpiLevelDelivered is either 1 or 2 and true is applicable only if SpiGeneral/SpiLevelDelivered is 0 or 2.

This is an implementation specific parameter.

Table 4-30. Attribute SpiPhyUnitSync (SpiPhyUnit) detailed description

Property	Value
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	true

4.4.17.4 SpiPhyHighSpeedMode (SpiPhyUnit)

Note

Specific if this HwUnit can join the high speed stransmission in Slave mode.

This is an implementation specific parameter.

Table 4-31. Attribute SpiPhyHighSpeedMode (SpiPhyUnit) detailed description

Property	Value
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.4.17.5 SpiPhyUnitClockRef (SpiPhyUnit)

Note

Reference to the SPI clock source configuration, which is set into the MCU driver configuration.

This is an implementation specific parameter.

Table 4-32. Attribute SpiPhyUnitClockRef (SpiPhyUnit) detailed description

Property	Value
Type	REFERENCE
Origin	Custom

4.4.17.6 SpiPhyUnitAlternateClockRef (SpiPhyUnit)

Note

Reference to the SPI alternate clock configuration, retrieved from the MCU plugin. Use to enable Spi_SetClockMode() function, which allows dual MCU clock configuration settings.

This is an implementation specific parameter.

Table 4-33. Attribute SpiPhyUnitAlternateClockRef (SpiPhyUnit) detailed description

Property	Value
Type	REFERENCE
Origin	Custom

4.4.17.7 SpiPhyUnitAsyncMethod (SpiPhyUnit)

Note

Transfer method of the periperal: DMA,PIO with FIFO.

This is an implementation parameter.

Table 4-34. Attribute SpiPhyUnitAsyncMethod (SpiPhyUnit) detailed description

Property	Value
Type	ENUMERATION
Origin	Custom
Symbolic Name	false
Default	PIO_FIFO

4.4.17.8 SpiPhyTxDmaChannel (SpiPhyUnit)

Note

DSPI Master Transmit DMA Logical Channel as configured by MCL plug-in, used to prepare the DSPI transmission dataframes starting from the TX buffer content.

This parameter is required only if SpiPhyUnitAsyncMethod - 'DMA' is selected.

This is an implementation specific parameter. The current DSPI TX source needs be configured for enabling this DMA channel.

Table 4-35. Attribute SpiPhyTxDmaChannel (SpiPhyUnit) detailed description

Property	Value
Type	CHOICE-REFERENCE
Origin	Custom

4.4.17.9 SpiPhyTxDmaChannelAux (SpiPhyUnit)

Note

DSPI Auxiliary Transmit DMA Logical Channel as configured by MCL plug-in, used to place dataframes into the DSPI registers.

This parameter is required only if SpiPhyUnitAsyncMethod - 'DMA' is selected.

This is an implementation specific parameter.

Table 4-36. Attribute SpiPhyTxDmaChannelAux (SpiPhyUnit) detailed description

Property	Value
Type	CHOICE-REFERENCE
Origin	Custom

4.4.17.10 SpiPhyRxDmaChannel (SpiPhyUnit)

Note

DSPI Receive DMA Logical Channel as configured by MCL plug-in, used to read the deserialized dataframes into the RX buffers.

This parameter is required only if SpiPhyUnitAsyncMethod - 'DMA' is selected.

This is an implementation specific parameter.

Table 4-37. Attribute SpiPhyRxDmaChannel (SpiPhyUnit) detailed description

Property	Value
Type	CHOICE-REFERENCE
Origin	Custom

4.4.18 Spi User Callback Header Files

Note

Spi User Callback Header Files.

Header file name which will be included by the Spi. The value of this parameter shall be used as h-char-sequence or q-char-sequence according to ISO C90 section 6.10.2 "source file inclusion". The parameter value MUST NOT represent a path, since ISO C90 does not specify how such a path is treated

Is included by form : Form SpiGeneral

4.5 Form SpiDemEventParameterRefs

Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.

Included forms :

- SPI_E_HARDWARE_ERROR

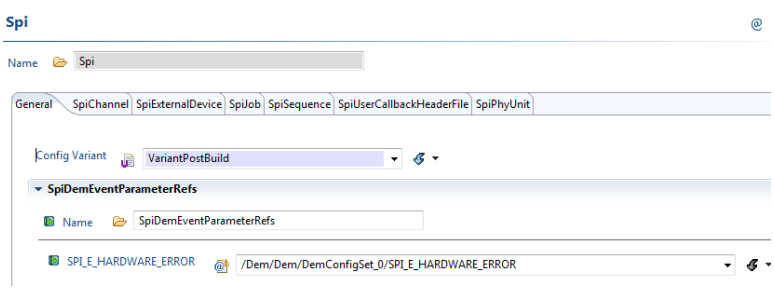

Figure 4-7. Tresos Plugin snapshot for SpiDemEventParameterRefs form.

4.5.1 SPI_E_HARDWARE_ERROR (SpiDemEventParameterRefs)

Table 4-38. Attribute SPI_E_HARDWARE_ERROR (SpiDemEventParameterRefs) detailed description

Property	Value
Type	SYMBOLIC-NAME-REFERENCE
Origin	AUTOSAR_ECUC

4.6 Form SpiNonAUTOSAR

Enabling the settings of this section will configure the driver in a mode not compliant with AUTOSAR requirements.

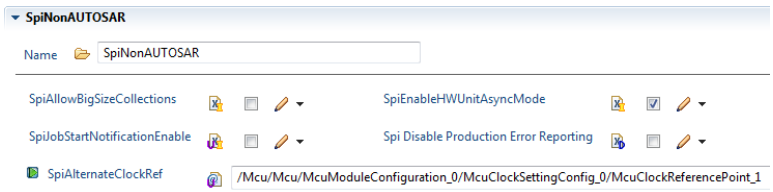

Figure 4-8. Tresos Plugin snapshot for SpiNonAUTOSAR form.

4.6.1 SpiEnableUserModeSupport (SpiNonAUTOSAR)

When this parameter is enabled, the Spi module will adapt to run from User Mode.

Note

Note: Spi module does not include registers protection. So, It is accessible to all registered in any public mode and does not affect by this field.

Table 4-39. Attribute SpiEnableUserModeSupport (SpiNonAUTOSAR) detailed description

Property	Value
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.6.2 SpiAllowBigSizeCollections (SpiNonAUTOSAR)

Note

A feature to allow more than 256 sequences, jobs, and channels.

Enabling this option will violate the following requirements: SPI166, SPI167, SPI168.

Table 4-40. Attribute SpiAllowBigSizeCollections (SpiNonAUTOSAR) detailed description

Property	Value
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.6.3 SpiEnableHWUnitAsyncMode (SpiNonAUTOSAR)

Note

Enable Spi_SetHWUnitAsyncMode() function, which allows defining distinct operation mode (POLLING or INTERRUPT) for each HWUnit.

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

Table 4-41. Attribute SpiEnableHWUnitAsyncMode (SpiNonAUTOSAR) detailed description

Property	Value
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.6.4 SpiTSBModeSupport (SpiNonAUTOSAR)

A feature to enable user to access the Micro Second Bus functionality.

Table 4-42. Attribute SpiTSBModeSupport (SpiNonAUTOSAR) detailed description

Property	Value
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.6.5 SpiITSBModeSupport (SpiNonAUTOSAR)

A feature to enable user to access the Interleave Micro Second Bus functionality.

Table 4-43. Attribute SpiITSBModeSupport (SpiNonAUTOSAR) detailed description

Property	Value
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.6.6 SpiJobStartNotificationEnable (SpiNonAUTOSAR)

Note

settings.

This feature is a non-Autosar feature to enable the start job notification.

Table 4-44. Attribute SpiJobStartNotificationEnable (SpiNonAUTOSAR) detailed description

Property	Value
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.6.7 SpiDisableDemReportErrorStatus (SpiNonAUTOSAR)

SpiDisableDemReportErrorStatus

Switches the Diagnostic Error Reporting and Notification OFF.

Table 4-45. Attribute SpiDisableDemReportErrorStatus (SpiNonAUTOSAR) detailed description

Property	Value
Label	Spi Disable Production Error Reporting
Type	BOOLEAN
Origin	Custom

Table continues on the next page...

Table 4-45. Attribute SpiDisableDemReportErrorStatus (SpiNonAUTOSAR) detailed description (continued)

Property	Value
Symbolic Name	false
Default	false

4.6.8 SpiAlternateClockRef (SpiNonAUTOSAR)

Note

Reference to the SPI alternate clock configuration, retrieved from the MCU plugin. Use to enable Spi_SetClockMode() function, which allows dual MCU clock configuration settings.

This is an implementation specific parameter.

Table 4-46. Attribute SpiAlternateClockRef (SpiNonAUTOSAR) detailed description

Property	Value
Type	REFERENCE
Origin	Custom

4.7 Form SpiDriver

This container holds the configuration of a single SPI Driver.

Included forms :

- Form SpiChannel
- Form SpiExternalDevice
- Form SpiJob
- Form SpiSequence

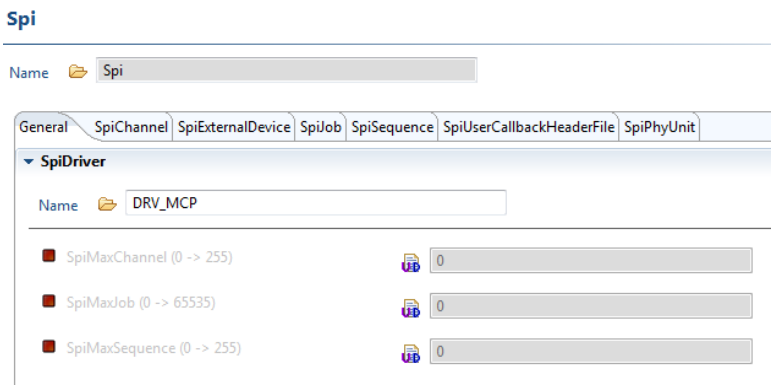

Figure 4-9. Tresos Plugin snapshot for SpiDriver form.

4.7.1 SpiMaxChannel (SpiDriver)

Note

This parameter contains the number of Channels configured. It will be gathered by tools during the configuration stage.

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

Table 4-47. Attribute SpiMaxChannel (SpiDriver) detailed description

Property	Value
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	0
Enable	false
Invalid	Range <=255 >=0

4.7.2 SpiMaxJob (SpiDriver)

Note

This parameter contains the number of Jobs configured. It will be gathered by tools during the configuration stage.

This parameter is not used, instead max jobs value is derived from number of jobs configured.

Table 4-48. Attribute SpiMaxJob (SpiDriver) detailed description

Property	Value
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	0
Enable	false
Invalid	Range <=65535 >=0

4.7.3 SpiMaxSequence (SpiDriver)

Note

This parameter contains the number of Sequences configured. It will be gathered by tools during the configuration stage.

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

Table 4-49. Attribute SpiMaxSequence (SpiDriver) detailed description

Property	Value
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	0
Enable	false
Invalid	Range <=255 >=0

4.7.4 Form SpiChannel

All data needed to configure one SPI-channel.

Is included by form : Form SpiDriver

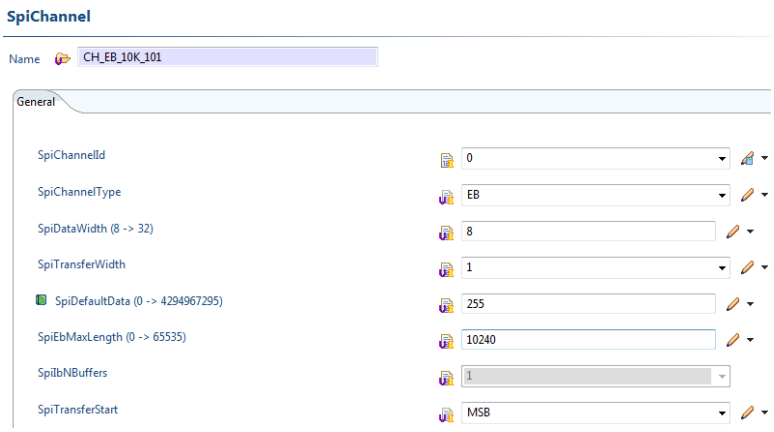

Figure 4-10. Tresos Plugin snapshot for SpiChannel form.

4.7.4.1 SpiChannelId (SpiChannel)

Channel ID of configured SPI channel.

Table 4-50. Attribute SpiChannelId (SpiChannel) detailed description

Property	Value
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	true
Invalid	Range <=255 >=0

4.7.4.2 SpiChannelType (SpiChannel)

Note

Buffer usage with EB/IB channel.

This parameter is dependant on SpiChannelBuffersAllowed parameter. When SpiChannelBuffersAllowed = 0; SpiChannelType should be IB When SpiChannelBuffersAllowed = 1; SpiChannelType should be EB When SpiChannelBuffersAllowed = 2; SpiChannelType can be IB or EB

Table 4-51. Attribute SpiChannelType (SpiChannel) detailed description

Property	Value
Type	ENUMERATION

Table continues on the next page...

Table 4-51. Attribute SpiChannelType (SpiChannel) detailed description (continued)

Property	Value
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	IB
Range	EB IB

4.7.4.3 SpiDataWidth (SpiChannel)

Note

This parameter is the width of a transmitted data unit.

The hardware supports data width from 4 to 16 bit.The unit is in bits. When SpiChannelBuffersAllowed = 0; SpiChannelType should be IB When SpiChannelBuffersAllowed = 1; SpiChannelType should be EB When SpiChannelBuffersAllowed = 2; SpiChannelType can be IB or EB

Table 4-52. Attribute SpiDataWidth (SpiChannel) detailed description

Property	Value
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	8
Invalid	Range <=16 >=4

4.7.4.4 SpiDefaultData (SpiChannel)

This parameter is the default value to transmit.

Table 4-53. Attribute SpiDefaultData (SpiChannel) detailed description

Property	Value
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	1

Table continues on the next page...

Table 4-53. Attribute SpiDefaultData (SpiChannel) detailed description (continued)

Property	Value
Invalid	Range <=4294967295 >=0

4.7.4.5 SpiEbMaxLength (SpiChannel)

This parameter contains the maximum size of data buffers in case of EB Channels and only.

Table 4-54. Attribute SpiEbMaxLength (SpiChannel) detailed description

Property	Value
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	1
Invalid	Range <=65535 >=0

4.7.4.6 SpiIbNBuffers (SpiChannel)

This parameter contains the maximum number of data buffers in case of IB Channels and only. In case of channel's Spi_DataWidth is 16, this parameter reffers to the number of bytes allocated to the buffers and MUST be even.

Table 4-55. Attribute SpiIbNBuffers (SpiChannel) detailed description

Property	Value
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	1
Invalid	Range <=65535 >=0

4.7.4.7 SpiEbSize (SpiChannel)

This parameter contains the number of frames can be stored in external buffer.

Table 4-56. Attribute SpiEbSize (SpiChannel) detailed description

Property	Value
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	1
Invalid	Range <=65535 >=0

4.7.4.8 SpiTransferStart (SpiChannel)

This parameter defines the first starting bit for transmission.

Table 4-57. Attribute SpiTransferStart (SpiChannel) detailed description

Property	Value
Type	ENUMERATION
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	MSB
Range	LSB MSB

4.7.5 Form SpiExternalDevice

The communication settings of an external device. Closely linked to SpiJob.

Is included by form : Form SpiDriver

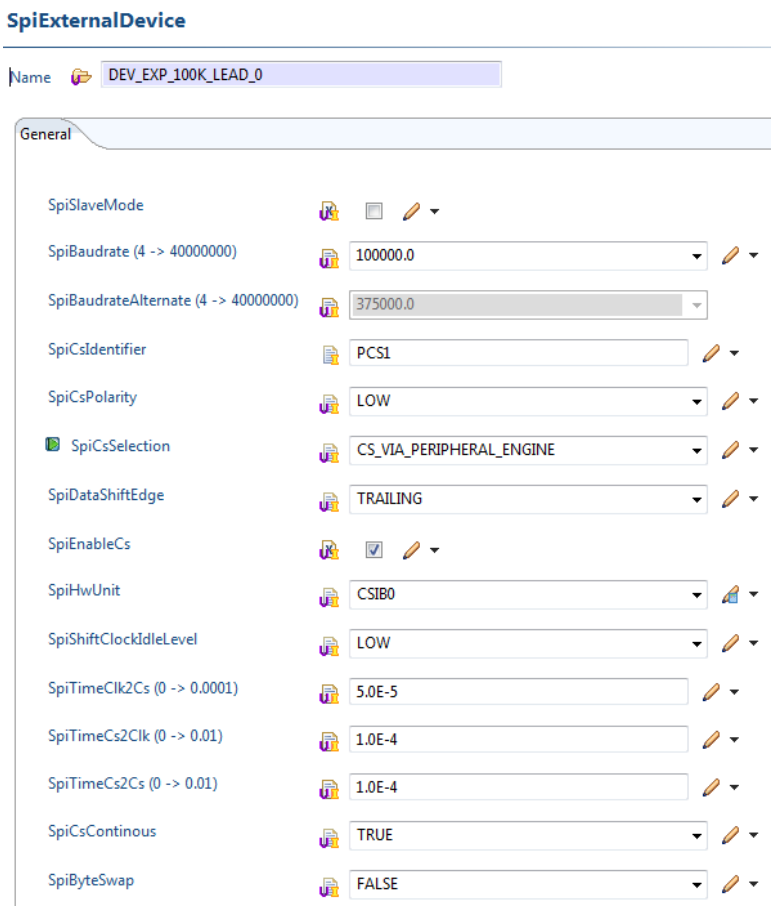

Figure 4-11. Tresos Plugin snapshot for SpiExternalDevice form.

4.7.5.1 SpiSlaveMode (SpiExternalDevice)

Note

Logical SpiHWunit to physical DSPI_[0|1|2|3|4|5] assignment. It depends on the number of units present in the chip version.

This is an implementation specific parameter.

Table 4-58. Attribute SpiSlaveMode (SpiExternalDevice) detailed description

Property	Value
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.7.5.2 SpiSlaveHighSpeedEnable (SpiExternalDevice)

Note

This is node which is used for enable the new feature in SPI Slave mode. The AutoSAR driver is Slave device and recevie the specific packages from External Master device. The sequence on Slave device will be called only once and never end.

This is an implementation specific parameter.

Table 4-59. Attribute SpiSlaveHighSpeedEnable (SpiExternalDevice) detailed description

Property	Value
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.7.5.3 TSBModeEnable (SpiExternalDevice)

Note

Specific if this HwUnit can only support timed serial bus mode. If true then this hardware unit is dedicated for TSB mode. If false then this hardware unit is dedicated for non TSB transfers only.

This is an implementation specific parameter.

Table 4-60. Attribute TSBModeEnable (SpiExternalDevice) detailed description

Property	Value
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.7.5.4 ITSBModeEnable (SpiExternalDevice)

Note

Specific if this HwUnit can only support timed serial bus mode. If true then this hardware unit is dedicated for ITSB mode. If false then this hardware unit is dedicated for non ITSB transfers only.

This is an implementation specific parameter.

Table 4-61. Attribute ITSBModeEnable (SpiExternalDevice) detailed description

Property	Value
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.7.5.5 SpiBaudrate (SpiExternalDevice)

This parameter is the communication baudrate - This parameter allows using a range of values, from the point of view of configuration tools, from Hz up to MHz. This field is used only in MASTER mode.

Note

The precision of this value depends SPI clock source configuration. If the driver cannot generate correct of the value, approximate value will be used.

Table 4-62. Attribute SpiBaudrate (SpiExternalDevice) detailed description

Property	Value
Type	FLOAT
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	100000
Invalid	Range <=40000000 >=4

4.7.5.6 SpiEnableCs (SpiExternalDevice)

This parameter enables or not the Chip Select handling functions. This parameter is closely linked to Job.If This parameter is True,then chip select is asserted and if False No chip select is asserted.

Table 4-63. Attribute SpiEnableCs (SpiExternalDevice) detailed description

Property	Value
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

4.7.5.7 SpiCsIdentifier (SpiExternalDevice)

This parameter is the symbolic name to identify the Chip Select (CS) allocated to this Job.

Table 4-64. Attribute SpiCsIdentifier (SpiExternalDevice) detailed description

Property	Value
Type	STRING
Origin	AUTOSAR_ECUC
Symbolic Name	true
Default	PCS0

4.7.5.8 SpiCsPolarity (SpiExternalDevice)

This parameter defines the active polarity of Chip Select.

Table 4-65. Attribute SpiCsPolarity (SpiExternalDevice) detailed description

Property	Value
Type	ENUMERATION
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	HIGH
Range	HIGH LOW

4.7.5.9 SpiCsSelection (SpiExternalDevice)

When the Chip select handling is enabled (see SpiEnableCs), then this parameter specifies if the chip select is handled automatically by Pe-ripheral HW engine or via general purpose IO by Spi driver.

Table 4-66. Attribute SpiCsSelection (SpiExternalDevice) detailed description

Property	Value
Type	ENUMERATION
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	CS_VIA_PERIPHERAL_ENGINE
Range	CS_VIA_PERIPHERAL_ENGINE CS_VIA_GPIO

4.7.5.10 SpiDataShiftEdge (SpiExternalDevice)

This parameter defines the SPI data shift edge.

Table 4-67. Attribute SpiDataShiftEdge (SpiExternalDevice) detailed description

Property	Value
Type	ENUMERATION
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	LEADING
Range	LEADING TRAILING

4.7.5.11 SpiHwUnit (SpiExternalDevice)

This parameter is the symbolic name to identify the HW SPI Hardware microcontroller peripheral allocated to this Job. CSIBn references the n-th logical unit configured in SpiPhyUnit container. For example: CSIB0 references the first logical unit (not the first DSPI_0 HW unit).

Table 4-68. Attribute SpiHwUnit (SpiExternalDevice) detailed description

Property	Value
Type	ENUMERATION

Table continues on the next page...

Table 4-68. Attribute SpiHwUnit (SpiExternalDevice) detailed description (continued)

Property	Value
Origin	AUTOSAR_ECUC
Symbolic Name	false

4.7.5.12 SpiShiftClockIdleLevel (SpiExternalDevice)

This parameter defines the SPI shift clock idle level.

Table 4-69. Attribute SpiShiftClockIdleLevel (SpiExternalDevice) detailed description

Property	Value
Type	ENUMERATION
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	HIGH
Range	HIGH LOW

4.7.5.13 SpiTimeClk2Cs (SpiExternalDevice)

Timing between clock and chip select in seconds (tASC) - This parameter allows to use a range of values from 0 up to 0.0001 Sec. The real configuration-value used in software BSW-SPI is calculated out of this by the generator-tools. If use continuous transfer(PCS signals remain asserted between transfers), tASC and tCSC will insert between transfers.

Table 4-70. Attribute SpiTimeClk2Cs (SpiExternalDevice) detailed description

Property	Value
Type	FLOAT
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	0.000001
Invalid	Range <=0.0001 >=0

4.7.5.14 SpiTimeCs2Clk (SpiExternalDevice)

Note

Timing between chip select and clock in seconds (tCSC) - This parameter allows to use a range of values from 0 up to 0.01 Sec. If use continuous transfer(PCS signals remain asserted between transfers), tASC and tCSC will insert between transfers.

This is an implementation specific parameter.

Table 4-71. Attribute SpiTimeCs2Clk (SpiExternalDevice) detailed description

Property	Value
Type	FLOAT
Origin	Custom
Symbolic Name	false
Default	0.000001
Invalid	Range <=0.01 >=0.0

4.7.5.15 SpiTimeCs2Cs (SpiExternalDevice)

Note

Timing between chip select assertions in seconds (tDT) - This parameter allows to use a range of values from 0 up to 0.01 Sec. If use continuous transfer(PCS signals remain asserted between transfers), tDT is not inserted between the transfers.

This is an implementation parameter.

Table 4-72. Attribute SpiTimeCs2Cs (SpiExternalDevice) detailed description

Property	Value
Type	FLOAT
Origin	Custom
Symbolic Name	false
Default	0.0000064
Invalid	Range <=64.0 >=0.0

4.7.5.16 SpiCsContinous (SpiExternalDevice)

Note

This field determines to keep chip select asserted between frame transfers.

This is an implementation parameter.

Table 4-73. Attribute SpiCsContinous (SpiExternalDevice) detailed description

Property	Value
Type	ENUMERATION
Origin	Custom
Symbolic Name	false
Default	TRUE
Range	TRUE FALSE

4.7.6 Form SpiJob

All data needed to configure one SPI-Job, amongst others the connection between the internal SPI unit and the special settings for an external device is done.

Is included by form : Form SpiDriver

Included forms :

- Form SpiChannelList

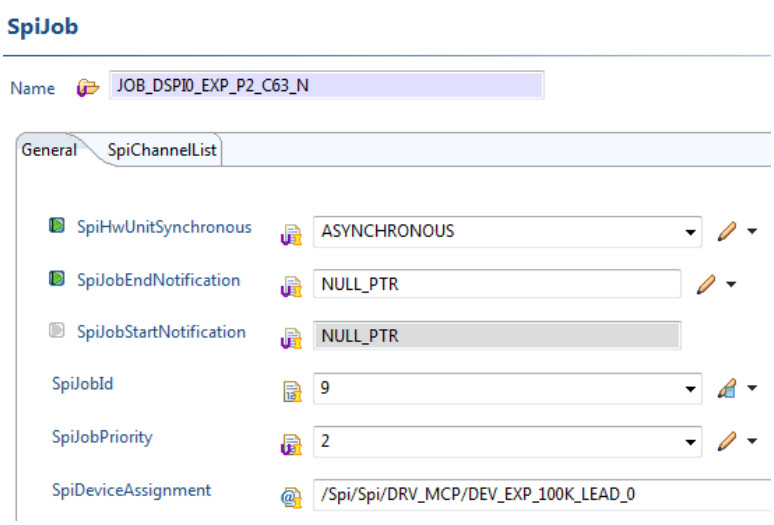

Figure 4-12. Tresos Plugin snapshot for SpiJob form.

4.7.6.1 TSBModeEnable (SpiJob)

Note

Specific if this HwUnit can only support timed serial bus mode. If true then this hardware unit is dedicated for TSB mode. If false then this hardware unit is dedicated for non TSB transfers only.

This is an implementation specific parameter.

Table 4-74. Attribute TSBModeEnable (SpiJob) detailed description

Property	Value
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.7.6.2 ITSBModeEnable (SpiJob)

Note

Specific if this HwUnit can only support timed serial bus mode. If true then this hardware unit is dedicated for ITSB mode. If false then this hardware unit is dedicated for non ITSB transfers only.

This is an implementation specific parameter.

Table 4-75. Attribute ITSBModeEnable (SpiJob) detailed description

Property	Value
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.7.6.3 SpiHwUnitSynchronous (SpiJob)

If SpiHwUnitSynchronous is set to "SYNCHRONOUS", the SpiJob uses its containing SpiDriver in a synchronous manner. If it is set to "ASYNCHRONOUS", it uses the driver in an asynchronous way. If the parameter is not set, the SpiJob uses the driver also in an asynchronous way.

Table 4-76. Attribute SpiHwUnitSynchronous (SpiJob) detailed description

Property	Value
Type	ENUMERATION
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	ASYNCHRONOUS
Enable	true
Range	ASYNCHRONOUS SYNCHRONOUS

4.7.6.4 SpiJobEndNotification (SpiJob)

This parameter is a reference to a notification function.

Table 4-77. Attribute SpiJobEndNotification (SpiJob) detailed description

Property	Value
Type	FUNCTION-NAME
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	NULL_PTR

4.7.6.5 SpiJobStartNotification (SpiJob)

This parameter is a reference to a notification function.

Table 4-78. Attribute SpiJobStartNotification (SpiJob) detailed description

Property	Value
Type	FUNCTION-NAME
Origin	Custom
Symbolic Name	false
Default	NULL_PTR

4.7.6.6 SpiJobId (SpiJob)

Job ID of configured SPI Job.

Table 4-79. Attribute SpiJobId (SpiJob) detailed description

Property	Value
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	true
Invalid	Range <=65535 >=0

4.7.6.7 SpiJobPriority (SpiJob)

Priority of the Job.

Table 4-80. Attribute SpiJobPriority (SpiJob) detailed description

Property	Value
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	0
Invalid	Range <=3 >=0

4.7.6.8 TSBFrameSize (SpiJob)

Note

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.

The hardware supports data width from 3 to 63 bit.The unit is in bits.

Table 4-81. Attribute TSBFrameSize (SpiJob) detailed description

Property	Value
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	16
Invalid	Range <=63 >=3

4.7.6.9 TS3_LEN (SpiJob)

Note

Time Slot 3 Length. This field provides the length of TS3 in terms of baud cycles. It must be ensured that the programmed value is greater than or equal to the length of the frame to be transmitted during TS3. Time Slot3 Length = TS3_LEN+1.

The hardware supports data length from 0 to 127 bit.This time slot shall be skipped if programmed to 0.

Table 4-82. Attribute TS3_LEN (SpiJob) detailed description

Property	Value
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	0
Invalid	Range <=127 >=0

4.7.6.10 TS2_LEN (SpiJob)

Note

Time Slot 2 Length. This field provides the length of TS2 in terms of baud cycles. It must be ensured that the programmed value is greater than or equal to the length of the frame to be transmitted during TS2. Time Slot2 Length = TS2_LEN+1.

The hardware supports data length from 0 to 127 bit.This time slot shall be skipped if programmed to 0.

Table 4-83. Attribute TS2_LEN (SpiJob) detailed description

Property	Value
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	0
Invalid	Range <=127 >=0

4.7.6.11 TS1_LEN (SpiJob)

Note

Time Slot 1 Length. This field provides the length of TS1 in terms of baud cycles. It must be ensured that the programmed value is greater than or equal to the length of the frame to be transmitted during TS1. Time Slot1 Length = TS3_LEN+1.

The hardware supports data length from 0 to 127 bit.This time slot shall be skipped if programmed to 0.

Table 4-84. Attribute TS1_LEN (SpiJob) detailed description

Property	Value
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	0
Invalid	Range <=127 >=0

4.7.6.12 TS0_LEN (SpiJob)

Note

Time Slot 0 Length. This field provides the length of TS0 in terms of baud cycles. It must be ensured that the programmed value is greater than or equal to the length of the frame to be transmitted during TS0. Time Slot0 Length = TS0_LEN+1.

The hardware supports data length from 0 to 127 bit.This time slot shall be skipped if programmed to 0.

Table 4-85. Attribute TS0_LEN (SpiJob) detailed description

Property	Value
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	0
Invalid	Range <=127 >=0

4.7.6.13 TS3_CONF (SpiJob)

This field is used to program the frame rule to be followed during Time Slot 3. This field is only valid when DSICR0[ITSB] mode is enabled

Table 4-86. Attribute TS3_CONF (SpiJob) detailed description

Property	Value
Type	STRING
Origin	Custom
Symbolic Name	false
Default	PF

4.7.6.14 TS2_CONF (SpiJob)

This field is used to program the frame rule to be followed during Time Slot 2. This field is only valid when DSICR0[ITSB] mode is enabled

Table 4-87. Attribute TS2_CONF (SpiJob) detailed description

Property	Value
Type	STRING
Origin	Custom
Symbolic Name	false
Default	PF

4.7.6.15 TS1_CONF (SpiJob)

This field is used to program the frame rule to be followed during Time Slot 1. This field is only valid when DSICR0[ITSB] mode is enabled

Table 4-88. Attribute TS1_CONF (SpiJob) detailed description

Property	Value
Type	STRING
Origin	Custom
Symbolic Name	false
Default	PF

4.7.6.16 TS0_CONF (SpiJob)

This field is used to program the frame rule to be followed during Time Slot 0. This field is only valid when DSICR0[ITSB] mode is enabled

Table 4-89. Attribute TS0_CONF (SpiJob) detailed description

Property	Value
Type	STRING
Origin	Custom
Symbolic Name	false
Default	PF

4.7.6.17 DsiCsIdentifier (SpiJob)

This parameter is the symbolic name to identify the Chip Select (CS) allocated to this Job in DSI mode.

Table 4-90. Attribute DsiCsIdentifier (SpiJob) detailed description

Property	Value
Type	STRING
Origin	Custom
Symbolic Name	false
Default	DPCS0

4.7.6.18 TransmitDataSource (SpiJob)

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.

Table 4-91. Attribute TransmitDataSource (SpiJob) detailed description

Property	Value
Type	ENUMERATION
Origin	Custom
Symbolic Name	false
Default	DSISerializationRegister

Table continues on the next page...

Table 4-91. Attribute TransmitDataSource (SpiJob) detailed description (continued)

Property	Value
Range	DSISerializationRegister DSIAltSerRegister

4.7.6.19 ChangeInDataTransfer (SpiJob)

This bit enables a change in serialization data to initiate DSI frames transfer. Only Continuous, and Change in data initiation control are supported.

Table 4-92. Attribute ChangeInDataTransfer (SpiJob) detailed description

Property	Value
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.7.6.20 DualReceiverSupport (SpiJob)

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 fo the frame.

Table 4-93. Attribute DualReceiverSupport (SpiJob) detailed description

Property	Value
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.7.6.21 SecondaryFrameSize (SpiJob)

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

Table 4-94. Attribute SecondaryFrameSize (SpiJob) detailed description

Property	Value
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	16
Invalid	Range <=31 >=3

4.7.6.22 SecondaryDsiCsIdentifier (SpiJob)

This parameter is the symbolic name to identify the Chip Select (CS) allocated to this Job in DSI mode in dual receiver mode.

Table 4-95. Attribute SecondaryDsiCsIdentifier (SpiJob) detailed description

Property	Value
Type	STRING
Origin	Custom
Symbolic Name	false
Default	DPCS0

4.7.6.23 SpiDeviceAssignment (SpiJob)

Reference to the external device used by this job.

Table 4-96. Attribute SpiDeviceAssignment (SpiJob) detailed description

Property	Value
Type	REFERENCE
Origin	AUTOSAR_ECUC

4.7.6.24 Form SpiChannelList

References to SPI channels and their order within the Job.

Is included by form : Form SpiJob

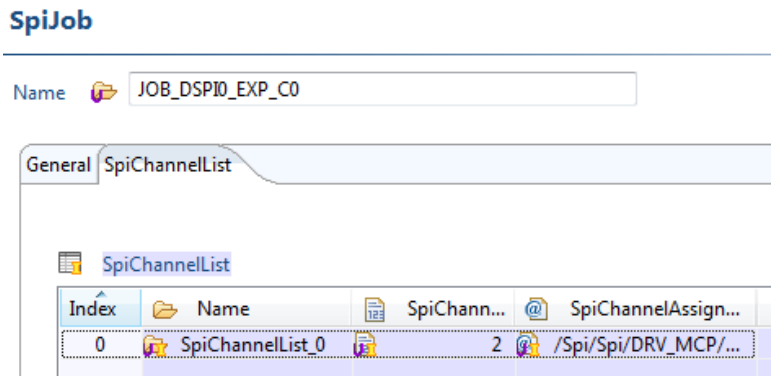

Figure 4-13. Tresos Plugin snapshot for SpiChannelList form.

4.7.6.24.1 SpiChannelIndex (SpiChannelList)

This parameter specifies the order of Channels within the Job.

Table 4-97. Attribute SpiChannelIndex (SpiChannelList) detailed description

Property	Value
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Invalid	Range <=255 >=0

4.7.6.24.2 SpiChannelAssignment (SpiChannelList)

A job references several channels.

Table 4-98. Attribute SpiChannelAssignment (SpiChannelList) detailed description

Property	Value
Type	REFERENCE
Origin	AUTOSAR_ECUC

4.7.7 Form SpiSequence

All data needed to configure one SPI-sequence.

Is included by form : Form SpiDriver

Included forms :

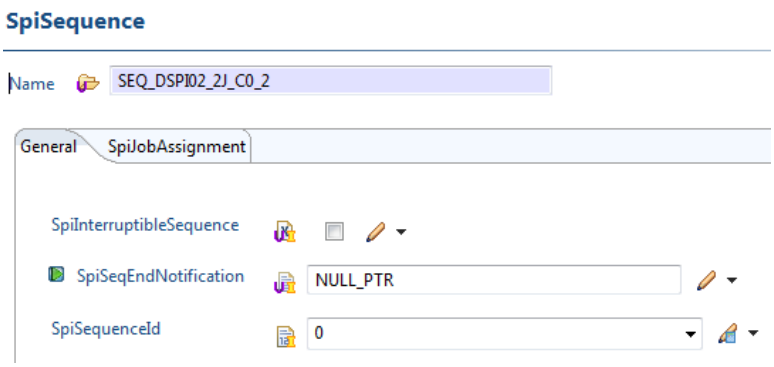

Figure 4-14. Tresos Plugin snapshot for SpiSequence form.

4.7.7.1 SpiInterruptibleSequence (SpiSequence)

This parameter allows or not this Sequence to be suspended by another one.

Table 4-99. Attribute SpiInterruptibleSequence (SpiSequence) detailed description

Property	Value
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.7.7.2 SpiSeqEndNotification (SpiSequence)

This parameter is a reference to a notification function.

Table 4-100. Attribute SpiSeqEndNotification (SpiSequence) detailed description

Property	Value
Type	FUNCTION-NAME
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	NULL_PTR

4.7.7.3 SpiSequenceId (SpiSequence)

Sequence ID of configured SPI Sequence.

Table 4-101. Attribute SpiSequenceId (SpiSequence) detailed description

Property	Value
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	true
Invalid	Range <=255 >=0

4.7.7.4 SpiJobAssignment (SpiJobAssignment)

A sequence references several jobs, which are executed during a communication sequence.

Table 4-102. Attribute SpiJobAssignment (SpiJobAssignment) detailed description

Property	Value
Type	REFERENCE
Origin	AUTOSAR_ECUC

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number UM35SPIASR4.2 Rev0002R1.0.0
Revision 1.0

AUTOSAR