

User Manual

for S32K1_S32M24X SPI Driver

Document Number: UM2SPIASRR21-11 Rev0000R2.0.0 Rev. 1.0

1 Revision History	2
2 Introduction	3
2.1 Supported Derivatives	3
2.2 Overview	4
2.3 About This Manual	5
2.4 Acronyms and Definitions	6
2.5 Reference List	6
3 Driver	8
3.1 Requirements	8
3.2 Driver Design Summary	8
3.3 Hardware Resources	10
3.4 Deviations from Requirements	11
3.5 Driver Limitations	14
3.5.1 Some LPSPI's features are not supported:	14
3.5.2 Limitation of LPSPI Slave mode:	14
3.5.3 Some limitations of SPI over FLEXIO:	15
3.6 Driver usage and configuration tips	15
3.6.1 How to use dual clock mode	15
3.6.2 How to configure handling Chip Select via general purpose IO (SpiCsSelection: CS_VIA_GPIO)	16
3.6.3 How to configure in multiple post build variants	18
3.6.4 How to configure Dma Fast transfer for a Sequence	19
3.6.5 How to configure SPI over FLEXIO	26
3.6.6 How to configure to use half duplex mode	31
3.6.7 How to use Half Duplex mode	33
3.6.8 Pointer to buffer alignment	35
3.7 Runtime errors	35
3.8 Symbolic Names Disclaimer	36
4 Tresos Configuration Plug-in	37
4.1 Module Spi	39
4.2 Container SpiDemEventParameterRefs	40
4.3 Reference SPI_E_HARDWARE_ERROR	40
4.4 Container SpiDriver	41
4.5 Parameter SpiMaxChannel	41
4.6 Parameter SpiMaxJob	42
4.7 Parameter SpiMaxSequence	42
4.8 Container SpiChannel	44
4.9 Parameter SpiChannelId	44
4.10 Parameter SpiChannelType	45

4.11 Parameter SpiDataWidth	46
4.12 Parameter SpiDefaultData	46
4.13 Parameter SpiEbMaxLength	47
4.14 Parameter SpiIbNBuffers	47
4.15 Parameter SpiTransferStart	48
4.16 Parameter SpiChannelHalfDuplexDirection	48
4.17 Reference SpiChannelEcucPartitionRef	49
4.18 Container SpiExternalDevice	50
4.19 Parameter SpiBaudrate	50
4.20 Parameter SpiCsIdentifier	51
4.21 Parameter SpiCsPolarity	52
4.22 Parameter SpiCsSelection	52
4.23 Parameter SpiDataShiftEdge	53
4.24 Parameter SpiEnableCs	53
4.25 Parameter SpiHwUnit	54
4.26 Parameter SpiShiftClockIdleLevel	54
4.27 Parameter SpiTimeClk2Cs	55
4.28 Parameter SpiTimeCs2Clk	55
4.29 Parameter SpiTimeCs2Cs	56
4.30 Parameter SpiCsBehavior	56
4.31 Parameter SpiDeviceHalfDuplexSupport	57
4.32 Parameter SpiTransferWidth	57
4.33 Parameter SpiHalfDuplexPinSelect	58
4.34 Reference SpiDeviceEcucPartitionRef	59
4.35 Container SpiJob	59
4.36 Parameter SpiJobEndNotification	60
4.37 Parameter SpiJobStartNotification	60
4.38 Parameter SpiJobId	61
4.39 Parameter SpiJobPriority	61
4.40 Reference SpiDeviceAssignment	62
4.41 Container SpiChannelList	62
4.42 Parameter SpiChannelIndex	62
4.43 Reference SpiChannelAssignment	63
4.44 Container SpiSequence	63
4.45 Parameter SpiInterruptibleSequence	64
4.46 Parameter SpiSeqEndNotification	64
4.47 Parameter SpiSequenceId	65
4.48 Parameter SpiEnableDmaFastTransfer	65
4.49 Reference SpiJobAssignment	66
4.50 Container SpiGeneral	66

4.51 Parameter SpiMulticoreSupport	67
4.52 Parameter SpiCancelApi	67
4.53 Parameter SpiChannelBuffersAllowed	68
4.54 Parameter SpiDevErrorDetect	68
4.55 Parameter SpiHwStatusApi	69
4.56 Parameter SpiInterruptibleSeqAllowed	69
4.57 Parameter SpiLevelDelivered	70
4.58 Parameter SpiMainFunctionPeriod	70
4.59 Parameter SpiSupportConcurrentSyncTransmit	72
4.60 Parameter SpiVersionInfoApi	72
4.61 Parameter SpiGlobalDmaEnable	73
4.62 Parameter SpiTimeoutMethod	73
4.63 Parameter SpiTransmitTimeout	74
4.64 Reference SpiEcucPartitionRef	75
4.65 Reference SpiKernelEcucPartitionRef	75
4.66 Container SpiPhyUnit	76
4.67 Parameter SpiPhyUnitMapping	76
4.68 Parameter SpiPhyUnitMode	77
4.69 Parameter SpiPhyUnitSync	77
4.70 Parameter SpiSamplePoint	78
4.71 Parameter SpiPinConfiguration	79
4.72 Parameter SpiPhyUnitAsyncUseDma	80
4.73 Parameter SpiMaxDmaFastTransfer	80
4.74 Reference SpiPhyUnitClockRef	81
4.75 Reference SpiPhyUnitAlternateClockRef	81
4.76 Reference SpiPhyTxDmaChannel	82
4.77 Reference SpiPhyRxDmaChannel	82
4.78 Reference SpiFlexioTxAndClkChannelsConfig	83
4.79 Reference SpiFlexioRxAndCsChannelsConfig	84
4.80 Container SpiPublishedInformation	84
4.81 Parameter SpiMaxHwUnit	85
4.82 Container CommonPublishedInformation	85
4.83 Parameter ArReleaseMajorVersion	86
4.84 Parameter ArReleaseMinorVersion	86
4.85 Parameter ArReleaseRevisionVersion	87
4.86 Parameter ModuleId	87
4.87 Parameter SwMajorVersion	88
4.88 Parameter SwMinorVersion	88
4.89 Parameter SwPatchVersion	89
4.90 Parameter VendorApiInfix	89

4.91 Parameter VendorId	90
4.92 Container SpiAutosarExt	90
4.93 Parameter SpiEnableUserModeSupport	91
4.94 Parameter SpiEnableDmaFastTransferSupport	91
4.95 Parameter SpiHalfDuplexModeSupport	92
4.96 Parameter SpiAllowBigSizeCollections	92
4.97 Parameter SpiEnableHWUnitAsyncMode	92
4.98 Parameter SpiJobStartNotificationEnable	93
4.99 Parameter SpiDisableDemReportErrorStatus	93
4.100 Parameter SpiFlexioEnable	94
5 Module Index	95
5.1 Software Specification	95
6 Module Documentation	96
6.1 Flexio_Spi Driver	96
6.1.1 Detailed Description	96
6.1.2 Data Structure Documentation	97
6.1.3 Macro Definition Documentation	100
6.1.4 Types Reference	100
6.1.5 Enum Reference	100
6.1.6 Function Reference	101
6.2 Lpspi Driver	108
6.2.1 Detailed Description	108
6.2.2 Data Structure Documentation	109
6.2.3 Types Reference	114
6.2.4 Enum Reference	114
6.2.5 Function Reference	116
6.2.6 Variable Documentation	125
6.3 Spi Driver	126
6.3.1 Detailed Description	126
6.3.2 Data Structure Documentation	130
6.3.3 Macro Definition Documentation	140
6.3.4 Types Reference	150
6.3.5 Enum Reference	151
6.3.6 Function Reference	154
6.3.7 Variable Documentation	166

Chapter 1

Revision History

Revision	Date	Author	Description
1.0	04.08.2023	NXP RTD Team	S32K1_S32M24X Real-Time Drivers AUTOSAR 4.4 & R21-11 Version 2.0.0

Chapter 2

Introduction

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

This User Manual describes NXP Semiconductor AUTOSAR Serial Peripheral Interface (SPI) for S32K1_S32M24X .

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 intended to be used with the following microcontroller devices of NXP Semiconductors:

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

- s32k142w_lqfp64
- s32k144_lqfp48
- s32k144_lqfp64 / MWCT1014S_lqfp64
- s32k144_lqfp100 / MWCT1014S_lqfp100
- s32k144_mapbga100
- s32k144w_lqfp48
- s32k144w_lqfp64
- s32k146_lqfp64
- s32k146_lqfp100 / MWCT1015S_lqfp100
- s32k146_mapbga100 / MWCT1015S_mapbga100
- s32k146_lqfp144
- s32k148_lqfp100
- s32k148_mapbga100 / MWCT1016S_mapbga100
- s32k148_lqfp144
- s32k148_lqfp176
- s32m241_lqfp64
- s32m242_lqfp64
- s32m243_lqfp64
- s32m244_lqfp64

All of the above microcontroller devices are collectively named as S32K1_S32M24X. Note: MWCT part numbers contain NXP confidential IP for Qi Wireless Power

2.2 Overview

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

AUTOSAR:

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

2.3 About This Manual

This Technical Reference employs the following typographical conventions:

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

Notes and warnings are shown as below:

Note

This is a note.

Warning

This is a warning

2.4 Acronyms and Definitions

Term	Definition
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSMI	Basic Software Make file Interface
CS	Chip Select
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DMA	Direct Memory Access
ECU	Electronic Control Unit
FIFO	First In First Out
LSB	Least Significant Bit
MCU	Micro Controller Unit
MIDE	Multi Integrated Development Environment
MSB	Most Significant Bit
N/A	Not Applicable
RAM	Random Access Memory
SIU	Systems Integration Unit
SWS	Software Specification
SPI	Serial Peripheral Interface
XML	Extensible Markup Language
BSW	Basic Software
ISR	Interrupt Service Routine
OS	Operating System
GUI	Graphical User Interface
PB Variant	Post Build Variant
PC Variant	Pre Compile Variant
LT Variant	Link Time Variant

2.5 Reference List

#	Title	Version
1	Specification of SPI Driver	AUTOSAR Release R21-11
2	S32K1 Series Reference Manual	Rev. 14, 09/2021
3	S32M24x Reference Manual	Rev. 2 Draft A, 05/2023
4	S32K1xx Data Sheet	Rev. 14, 08/2021
5	S32M2xx Data Sheet	Rev. 3 DraftA — 05/2023
6	Errata S32K116_0N96V	Rev. 22/OCT/2021
7	Errata S32K118_0N97V	Rev. 22/OCT/2021
8	Errata S32K142_0N33V	Rev. 22/OCT/2021
9	Errata S32K144_0N57U	Rev. 22/OCT/2021
10	Errata S32K144W_0P64A	Rev. 22/OCT/2021

#	Title	Version
11	Errata S32K146_0N73V	Rev. 22/OCT/2021
12	Errata S32K148_0N20V	Rev. 22/OCT/2021
13	S32M242_N33V+P73G	Rev. 0, 6/2023
14	S32M244_P64A+P73G	Rev. 0

Chapter 3

Driver

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

3.1 Requirements

Requirements for this driver are detailed in the AUTOSAR R21-11 Rev0000 SPI Driver Software Specification document (See Table [Reference List](#)).

AUTOSAR deviations from requirements are described in [Deviations from Requirements](#) chapter of this document..

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 SpiPhyUnitAlternateClockRef) 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: Lpspi_Ip_LPSPi_X_IrqTxDmaHandler, Flexio_Spi_Ip_FLEXIO_SPI_X_IrqTxDmaHandler and Lpspi_Ip_LPSPi_X_IrqRxDmaHandler, Flexio_Spi_Ip_FLEXIO_SPI_0_IrqRxDmaHandler where X is the SPI unit used (eg: Lpspi_Ip_LPSPi_0_IrqTxDmaHandler if LPSPi0 is used).

Interrupt request usage

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

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

```
<ISR_function_name()>
```

```
#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.

SPI Physical Units: Has a different on number instance between each derivatives of the chip S32K1XX

- Only S32K116 has LPSPI_0, FLEXIO_SPI_0 and FLEXIO_SPI_1 hardware instances.
- Both S32K118 and S32K142 have LPSPI_0, LPSPI1, FLEXIO_SPI_0 and FLEXIO_SPI_1 hardware instances.
- And the rest S32k142w, S32K144, S32K144w, S32K146 and S32K148 have LPSPI_0, LPSPI_1, LPSPI_2, FLEXIO_SPI_0, FLEXIO_SPI_1 hardware instances.

Note

In EB tresos, SPI Physical Unit has selected by SpiPhyUnitMapping.

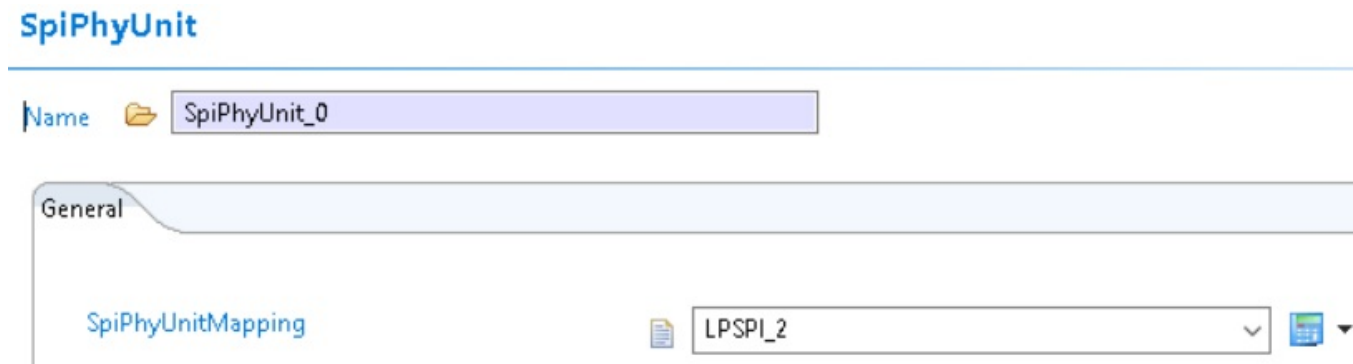


Figure 3.1 SPI Physical Unit has selected by SpiPhyUnitMapping in EB Tresos

For example with the chip S32K1XX:

The LPSPI_0, LPSPI_1 and LPSPI_2 use the pins correspondingly with naming is LPSPI0, LPSPI1, LPSPI2. The FLEXIO_SPI_0 and FLEXIO_SPI_1 use the pins correspondingly with naming is FXIO. The pins can find in the file IO Signal Description for each derivatives (e.g. "S32K148_IO_Signal_Description_Input_Multiplexing.xlsx" from attached file of S32K1XX Reference Manual Reference List.

LPSPI1 can be found in the xlsx file with naming is LPSPI1. And the Pin-Muxing is:

LPSPI1	LPSPI1_PCS0	1	0000_0011	IO_PAD	PTD3	D10	101	122	70
		2	0000_0101	IO_PAD	PTE1	A4	137	165	93
		3	0000_0100	IO_PAD	PTA21			6	
		4	0000_0011	IO_PAD	PTA26		19	26	
		5	-	-	disable low				
LPSPI1	LPSPI1_PCS1	1	0000_0011	IO_PAD	PTA6	G9	85	104	58
		2	0000_0100	IO_PAD	PTB18		36	43	
		3	0000_0100	IO_PAD	PTA22			9	
		4	-	-	disable low				
LPSPI1	LPSPI1_SCK	1	0000_0011	IO_PAD	PTD0	C2	4	8	4
		2	0000_0011	IO_PAD	PTB14	F10	96	116	66
		3	0000_0100	IO_PAD	PTA19			2	
		4	0000_0011	IO_PAD	PTA28		24	31	
		5	-	-	disable low				
LPSPI1	LPSPI1_SIN	1	0000_0011	IO_PAD	PTB15	F9	95	115	65
		2	0000_0011	IO_PAD	PTD1	B1	3	7	3
		3	0000_0100	IO_PAD	PTA20			3	
		4	0000_0101	IO_PAD	PTA29		26	33	
		5	-	-	disable low				
LPSPI1	LPSPI1_SOUT	1	0000_0011	IO_PAD	PTD2	D9	102	123	71
		2	0000_0011	IO_PAD	PTB16	F8	94	114	64
		3	0000_0101	IO_PAD	PTE0	B4	138	166	94
		4	0000_0100	IO_PAD	PTA18			1	
		5	0000_0011	IO_PAD	PTA27		22	29	
		6	-	-	disable low				

Figure 3.2 IO Signal Description for LPSPI_1

3.4 Deviations from Requirements

The driver deviates from the AUTOSAR Spi Driver software specification in some places. The table identifies the AUTOSAR requirements that are not fully implemented, implemented differently, not available, not testable or out of scope for the Spi Driver.

Term	Definition
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, not available, not testable or out of scope for the driver.

Table 3.2 Driver Deviations Table

Requirement	Status	Description	Notes
SWS_Spi_00040	N/S	The SPI Handler/Driver handles only the Master mode.	Both Master and Slave are supported by SPI driver.
SWS_Spi_00342	N/S	Depending on microcontrollers, the SPI peripheral could share registers with other peripherals. In this typical case, the SPI Handler/Driver has a relationship with MCU module [REF] for initialising and de-initialising those registers.	There is no register that is shared with other peripherals.

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.
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 synchronous transmit. Please see also SW↔S_Spi_00267 and SWS_Spi_↔00384.	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.

Requirement	Status	Description	Notes
SWS_Spi_00383	N/S	<p>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)</p>	<p>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.</p>
SWS_Spi_00385	N/S	<p>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)</p>	<p>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.</p>

Requirement	Status	Description	Notes
SWS_Spi_00386	N/S	When no error bit inside the S←PI transmit status register is set, the SPI transmit status register information shall be reported to DEM as Dem_ReportError←Status (SPI_E_HARDWAR←E_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_Async←Transmit 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.
SWS_Spi_00999	N/S	These requirements are not applicable to this specification.	This is not a requirement.
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 parameter CS←_VIA_GPIO, the user has to use SpiJobStartNotification' & 'SpiJobEndNotification' to toggle the CS (chip select pin) using DIO drivers .
SWS_Spi_CONSTR_00001	N/S	DRATF: The ECUC partitions referenced by SpiKernelEcuc←PartitionRef shall be a subset of the ECUC partitions referenced by SpiEcucPartitionRef.	Type IV Autosar multicore not implemented for current module. Multicore Type II is implemented. AAI-445.

3.5 Driver Limitations

3.5.1 Some LPSPI's features are not supported:

- Host request can be used to control the start of a SPI bus transfer
- Receive data match logic supporting wakeup on data match
- Half duplex feature in receive mode doesn't support above 32 bits transfer and continuous CS.

3.5.2 Limitation of LPSPI Slave mode:

- In Slave mode and use interrupt mode without DMA, the application needs to make sure Slave's interrupt service is not delayed to avoid errors underflow and overflow occur.

3.5.3 Some limitations of SPI over FLEXIO:

- Do not support CS Non-continuous mode.
- Do not support the polarity of CS is HIGH (only LOW is supported).
- "SpiCsIdentifier" (in SpiExternalDevice) is not used to select the no. of CS. It will be configured by "SpiFlexioRxAndCsChannelsConfig" in "SpiPhyUnit". Please refer "How to configure SPI over FLEXIO" section for more details
- SPI FLEXIO does not support to configure the timing delay.
- In DMA mode, Spi over FLEXIO does not support MSB mode in both transmission and reception modes except data width differs to 8, 16 or 32 bits.
- Spi over FLEXIO does not support above 32 bits transfer.

3.6 Driver usage and configuration tips

3.6.1 How to use dual clock mode

This mode permits to change the clock reference(referred by the field SpiPhyUnitAlternateClockRef) and to keep the basic characteristics of the transmission(like baudrate, delay time). The user can configuration two McuClockSettingConfig are McuClockSettingConfig_Normal and McuClockSettingConfig_Alternate:

Index	Name	Mcu Clock	FXOSC u...	FXOSC E...	FXOSC B...	FXOSC C...	FXOSC E...	Crystal o...	FXOSC
0	McuClockSettingConfig_Normal	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	100	0	4
1	McuClockSettingConfig_Alternate	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	157	12	4

Figure 3.3 Configuration two McuClockSettingConfig

The field SpiPhyUnitClockRef will be referred to McuClockSettingConfig_Normal.

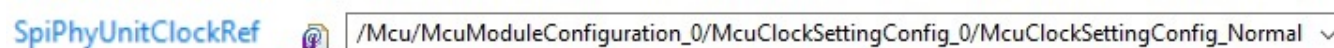


Figure 3.4 SpiPhyUnitClockRef will be referred to McuClockSettingConfig_Normal

The field SpiPhyUnitAlternateClockRef will be referred to McuClockSettingConfig_Alternate.

 SpiPhyUnitAlternateClockRef @ /Mcu/Mcu/McuModuleConfiguration_0/McuClockSettingConfig_0/McuClockSettingConfig_Alternate

Figure 3.5 SpiPhyUnitAlternateClockRef will be referred to McuClockSettingConfig_Alternate

The default, Spi driver will use Normal Clock with the clock reference by the field SpiPhyUnitClockRef. If the user changes the clock setting to McuClockSettingConfig_Alternate by the function Mcu_InitClock(McuClockSettingConfig_Alternate), the user also changes clock mode for Spi driver by the function Spi_SetClockMode(SPI_ALTERNATE).

3.6.2 How to configure handling Chip Select via general purpose IO (SpiCsSelection: CS_VIA_GPIO)

The chip select is handled automatically by Pe-ripheral HW engine or via general purpose IO by Spi driver. In the case of the hardware does not support to keep chip select asserted between frame transfers. The user can use the CS_VIA_GPIO feature (selected by node SpiCsSelection) and the driver will call functions notification (defined by nodes SpiJobStartNotification and SpiJobEndNotification) to control CS pin via GPIO for each Job. By this way, SPI driver can communication with external device requires Continuous CS.

The configuration steps use the CS_VIA_GPIO feature as below:

- Select the CS_VIA_GPIO feature by node SpiCsSelection

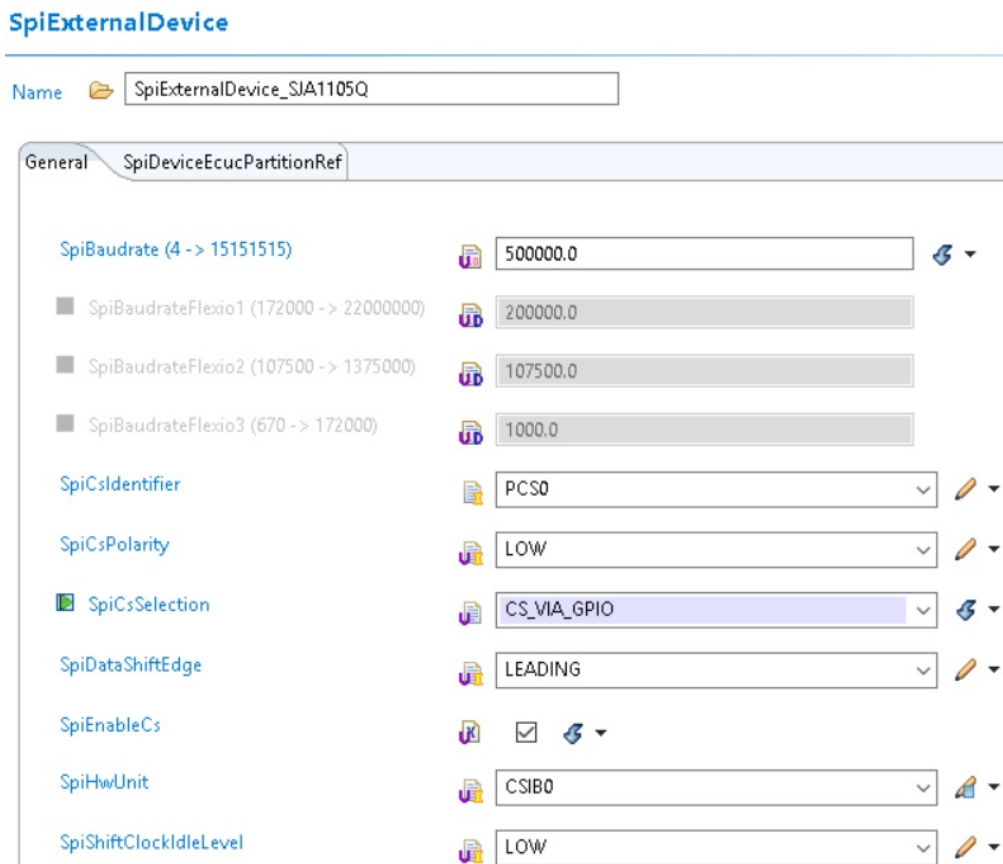


Figure 3.6 Select the CS_VIA_GPIO feature

- Enable Job Start Notification by node SpiJobStartNotificationEnable

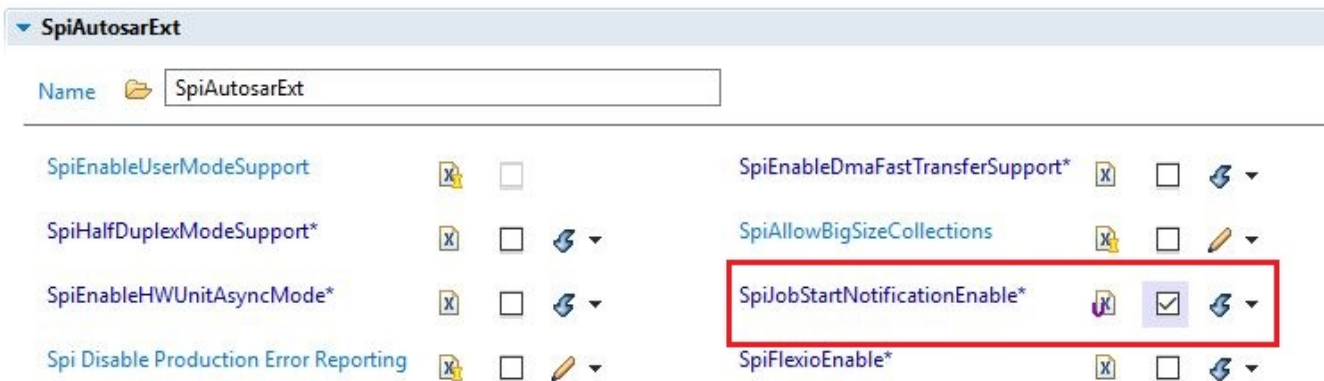



Figure 3.7 Enable Job Start Notification on SpiAutosarExt form

- Define two functions Job Start Notification and Job End Notification. Ex: Start_Job_Notification_Function, End_Job_Notification_Function. Start_Job_Notification_Function function will must assert CS via GPIO. End_Job_Notification_Function will must de-assert CS via GPIO. After that, enter name of two functions on nodes SpiJobStartNotification and SpiJobEndNotification.

SpiJob

Name  SpiJob_0

General SpiChannelList












 SpiJobEndNotification	 Start_Job_Notification_Function	 ▼
 SpiJobStartNotification	 End_Job_Notification_Function	 ▼
SpiJobId	 0	 ▼
SpiJobPriority	 0	 ▼
SpiDeviceAssignment	 /Spi/Spi/EthSwt_SpiDriver/SpiExternalDevice_0	

Figure 3.8 Enter name of two functions SpiJobStartNotification and SpiJobEndNotification on SpiJob form

3.6.3 How to configure 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 be the same between all post build variants. Names and Index of ExternalDevices must be 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 have 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_DSPIO_1J_C0_1 and SpiSequenceId is 0.
- Name of job is JOB_DSPIO_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_DSPIO_1J_C0_1 0
#define SpiConf_SpiSequence_SEQ_DSPI1_1J_C0_2 1
#define SpiConf_SpiJob_JOB_DSPIO_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.

3.6.4 How to configure Dma Fast transfer for a Sequence

The SPI driver can be able to transfer a Sequence with multiple Channels, Jobs without any CPU intervention between the channels and jobs. The CPU is used only to start the transmission (Spi_AsyncTransmit) and to process end of sequence notification. The driver will use DMA Scatter/Gather feature for all 2 DMA channels SpiPhyTxDmaChannel and SpiPhyRxDmaChannel. SpiPhyTxDmaChannel will fill data to TCR and TDR registers. SpiPhyRxDmaChannel will read data from RDR register.

This feature requires:

- The parameters SpiBaudrate, SpiHwUnit, SpiTimeClk2Cs, SpiTimeCs2Clk, SpiTimeCs2Cs in External Device linked to each Job in this Sequence must be the same.
- The parameters SpiDataWidth and SpiTransferStart in Channel assigned to each Job in this Sequence must be the same.

Driver

- In each Channel, the number of data buffers is NOT higher than 32767 if SpiDataWidth < 9. So, SpiIbNBuffers and SpiEbMaxLength must be assigned to suitable values.
- Only Master mode is supported(SpiPhyUnitMode = SPI_MASTER).
- Make sure that SpiMaxDmaFastTransfer value in SpiPhyUnit allocated to this Sequence must NOT lower than total of Channels in this Sequence.
- Make sure that number of ScatterGathers configuration in each SpiPhyUnit/SpiPhyTxDmaChannel must NOT lower than (total of Channels * 2) plus total of Jobs in this Sequence.
- Make sure that number of ScatterGathers configuration in each SpiPhyUnit/SpiPhyRxDmaChannel must NOT lower than total of Channels in this Sequence.
- Only SpiJobStartNotification and SpiJobEndNotification can be supported for first Job in a Sequence. But SpiJobEndNotification will be called at the end of Sequence as SpiSeqEndNotification.

Configuration example for 4 Channels and 2 Jobs in Sequence:

- Enable Dma Fast transfer support.

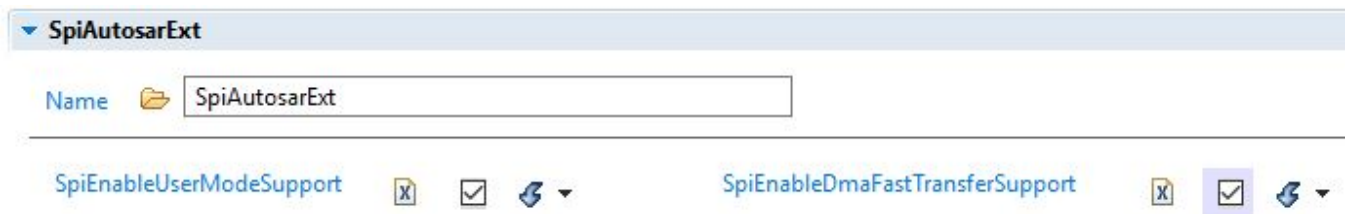



Figure 3.9 Enable Dma Fast transfer support

- Enable Dma Fast transfer for Sequence.

SpiSequence

Name  SEQ_DSP11_EXP_2J_C71_67_N

General SpiJobAssignment

SpiInterruptibleSequence



 SpiSeqEndNotification



EndSeq_DSP11_EXP_2J_C71_67



SpiSequenceld



0




SpiEnableDmaFastTransfer



Figure 3.10 Enable Dma Fast transfer for Sequence

- Enable Dma mode and set maximum number of Dma Fast transfers session. Make sure that the value of SpiMaxDmaFastTransfer is higher or equal total number of Channels in a Sequence.

SpiPhyUnit

Name  SpiPhyUnit_0

General
















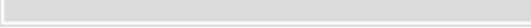










SpiPhyUnitMapping	 LPSP1_2  
SpiPhyUnitMode	 SPI_MASTER  
SpiPhyUnitSync	 <input type="checkbox"/> 
SpiSamplePoint (0 -> 1)	 0 
SpiPinConfiguration (0 -> 3)	 0 
SpiPhyUnitClockRef	 /Mcu/Mcu/McuModuleConfiguration/McuClockSettingCor
 SpiPhyUnitAlternateClockRef	 
SpiPhyUnitAsyncUseDma	 <input checked="" type="checkbox"/> 
 SpiPhyTxDmaChannel	 /Mcl/Mcl/MclConfig/DMA_LPSP12_TX
 SpiPhyRxDmaChannel	 /Mcl/Mcl/MclConfig/DMA_LPSP12_RX
 SpiMaxDmaFastTransfer	 4 



Figure 3.11 Enable Dma mode and set maximum number of Dma Fast transfers session



- Configure for SpiPhyTxDmaChannel in MCL.



Logic Channel



Name  DMA_LPSPi2_TX



Logic Channel Configuration Global Transfer ScatterGather



Logic Channel Name  DMA_LOGIC_CH_0 




Hardware Instance  DMA_IP_HW_INST_0 

Hardware Channel  DMA_IP_HW_CH_0 

Interrupt Callback  Lpspi_Ip_LPSPi2_IrqTxDmaHandler 

Error Interrupt Callback  NULL_PTR 

 Ecuc Partition Ref 

Enable Global Config  ☒  Enable Transfer Config  ☐




Enable Scatter/Gather  ☒ 


Figure 3.12 Configure for SpiPhyTxDmaChannel in MCL

Logic Channel


Name  DMA_LPSPi2_TX





Logic Channel Configuration Global Transfer ScatterGather

▼ dmaLogicChannel_GlobalConfigType


Name  dmaLogicChannel_GlobalConfigType





▼ Control



Name  dmaLogicChannelConfig_GlobalControlType



Enable Master Id Replication  ☐  Enable Buffered Writes  ☐ 

▼ Request

Name  dmaLogicChannelConfig_GlobalRequestType

Enable DMAMUX Trigger  ☐  Enable DMAMUX Source  ☒ 

DMAMUX0 Source  DMA_IP_REQ_MUX0_LPSPi2_TX 

DMAMUX1 Source  DMA_IP_REQ_MUX1_DISABLED 




Enable DMA Request  ☐ 


Figure 3.13 Global configuration for SpiPhyTxDmaChannel in MCL

Number of ScatterGather elements will equal to SpiMaxDmaFastTransfer * 2 + total number of Jobs in Sequence.

Logic Channel

Name  DMA_LPSP12_TX

Logic Channel Configuration Global Transfer ScatterGather

 ScatterGather












Index	Name	Element Name	Last Element of the Link
0	TXSG0	DMA_LOGIC_CH_0_SGA_ELEMENT_0	 <input type="checkbox"/>
1	TXSG1	DMA_LOGIC_CH_0_SGA_ELEMENT_1	 <input type="checkbox"/>
2	TXSG2	DMA_LOGIC_CH_0_SGA_ELEMENT_2	 <input type="checkbox"/>
3	TXSG3	DMA_LOGIC_CH_0_SGA_ELEMENT_3	 <input type="checkbox"/>
4	TXSG4	DMA_LOGIC_CH_0_SGA_ELEMENT_4	 <input type="checkbox"/>
5	TXSG5	DMA_LOGIC_CH_0_SGA_ELEMENT_5	 <input type="checkbox"/>
6	TXSG6	DMA_LOGIC_CH_0_SGA_ELEMENT_6	 <input type="checkbox"/>
7	TXSG7	DMA_LOGIC_CH_0_SGA_ELEMENT_7	 <input type="checkbox"/>
8	TXSG8	DMA_LOGIC_CH_0_SGA_ELEMENT_8	 <input type="checkbox"/>
9	TXSG9	DMA_LOGIC_CH_0_SGA_ELEMENT_9	 <input checked="" type="checkbox"/>

Figure 3.14 ScatterGather configuration for SpiPhyTxDmaChannel in MCL


"Element Link" node must be configured for each ScatterGather element as a continuation sequence: TXSG0->TXSG1->TXSG2->TXSG3->TXSG4->TXSG5->TXSG6->TXSG7->TXSG8->TXSG9.


ScatterGather Element



Name  TXSG0


General

Element

Name  dmaLogicChannelConfig_ScatterGatherElementConfigType

Element Name  DMA_LOGIC_CH_0_SGA_ELEMENT_0

Last Element of the Link  ☐ 

Element Link  /Mcl/Mcl/MclConfig/DMA_LPSP12_TX/dmaLogicChannel_ConfigType/dmaLogicChannel_ScatterGatherConfigType/TXSG1



Enable Scatter/Gather Config  ☐ 

Figure 3.15 Link each ScatterGather element as a continuation sequence in MCL

- Configure for SpiPhyRxDmaChannel in MCL is similar to SpiPhyTxDmaChannel. But number of Scatter↔Gather elements will equal to SpiMaxDmaFastTransfer.

Logic Channel

The screenshot shows the 'Logic Channel Configuration' window in the MCL tool. The 'Name' field is set to 'DMA_LPSPi2_RX'. The 'Logic Channel Configuration' tab is active, showing the following settings:

- Logic Channel Name:** DMA_LOGIC_CH_2
- Hardware Instance:** DMA_IP_HW_INST_0
- Hardware Channel:** DMA_IP_HW_CH_2
- Interrupt Callback:** Lpspi_lp_LPSPi2_IrqRxDmaHandler
- Error Interrupt Callback:** NULL_PTR
- Ecuc Partition Ref:** (Empty)
- Enable Global Config:** ☒
- Enable Transfer Config:** ☐
- Enable Scatter/Gather:** ☒

Figure 3.16 Configure for SpiPhyRxDmaChannel in MCL

- Notice that "Level Priority" value of SpiPhyRxDmaChannel is higher than SpiPhyTxDmaChannel. However, inside Platform module, the "Level Priority" value should be Tx DMA ISR higher than Rx DMA ISR.

3.6.5 How to configure SPI over FLEXIO

First of all, SPI over FLEXIO must use MCL module to enable/disable Flexio module and configure Flexio Channels (Select PIN, Shifter, Timer registers) and ISR handler. SPI can not work by itself. Bellowing is guiding to configure SPI over FLEXIO for both EB tresos and CT configuration tool

Note

Mcl_Init() functions must be called before using SPI over FLEXIO.

3.6.5.1 EB Tresos

- Enable flexio mode first by set SpiFlexioEnable: (bellow picture)

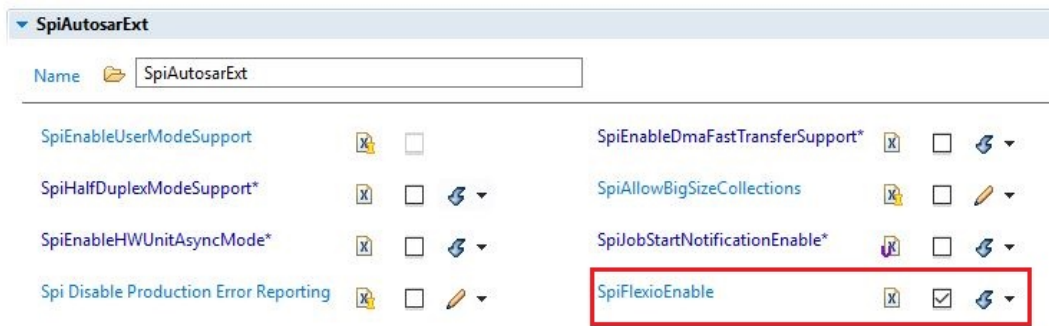


Figure 3.17 Enable SPI over FLEXIO

- Enable Flexio support on MCL site: (bellow picture)



Figure 3.18 Enable Mcl FLEXIO support

- In MCL, setup a set for SPI FLEXO Channels like: (bellow picture)

- Flexio Channel

is no. of Shifter and Timer registers will be selected for each FLEXIO channel. For example: CHANNEL_0 selected means Shifter_0 and Timer_0 will be selected and locked, other channels cannot select those registers.

- Flexio Pin

is no. of PIN will be selected for each FLEXIO channel. For example: PIN_0 selected means FXIO_D0 will be selected and locked, other channels cannot select this PIN.

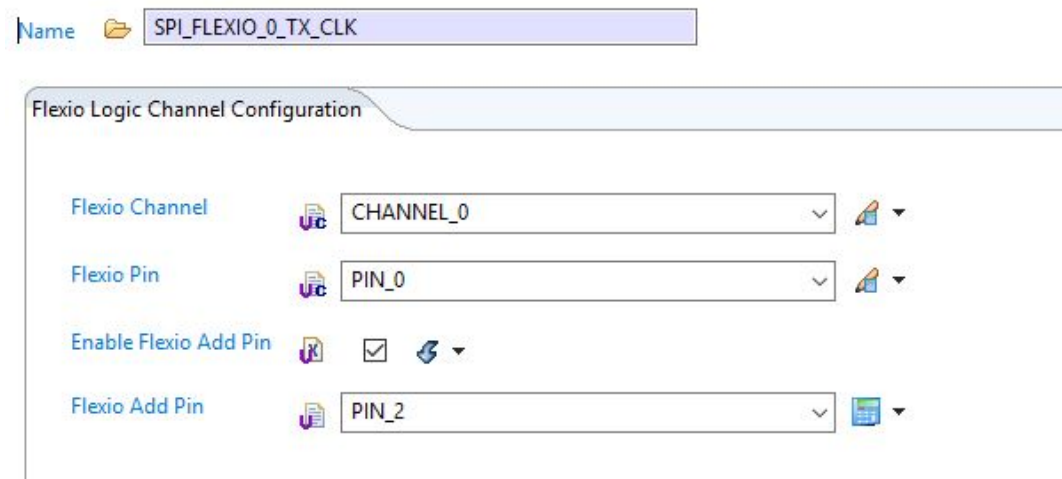


Figure 3.19 A set of FLEXIO SPI channels from MCL

- From SPI, each FLEXIO_SPI_x is selected in SpiPhyUnit then those set must be enabled and selected correctly value (refer from MCL which was configured above) (bellow picture)
 - SpiFlexioTxAndClkChannelsConfig: FLEXIO Logical Channel for TX(MOSI) and CLK(Clock). This selects 1 PIN to setup as TX(MOSI) channel and 1 PIN to set up as CLK channel. If you select CHANNEL_X, PIN_Y(FlexioMclPinId) and PIN_Z(FlexioMclAddPinId) that means:
 - * ShifterX is selected for TX(MOSI) channel
 - * TimerX is selected for CLK channel.
 - * PIN_Y is selected for TX(MOSI) channel
 - * PIN_Z is selected for CLK channel
 - SpiFlexioRxAndCsChannelsConfig: FLEXIO Logical Channel for RX(MISO) and CS(Chip select). This selects 1 PIN to setup as MISO channel and 1 PIN to set up as CS channel. If you select CHANNEL_X, PIN_Y(FlexioMclPinId) and PIN_Z(FlexioMclAddPinId) that mean:
 - * ShifterX is selected for RX(MISO) channel
 - * TimerX is selected for CS channel.
 - * PIN_Y is selected for RX(MISO) channel
 - * PIN_Z is selected for CS channel

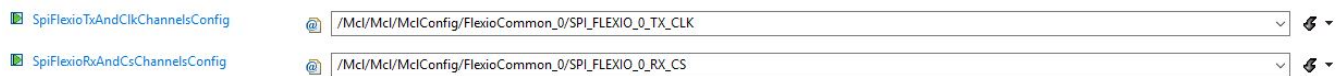


Figure 3.20 A set of FLEXIO SPI channels from SPI

3.6.5.2 Design Studio Tool:

It is the same with EB tresos

- Enable flexio mode first by set SpiFlexioEnable: (bellow picture)

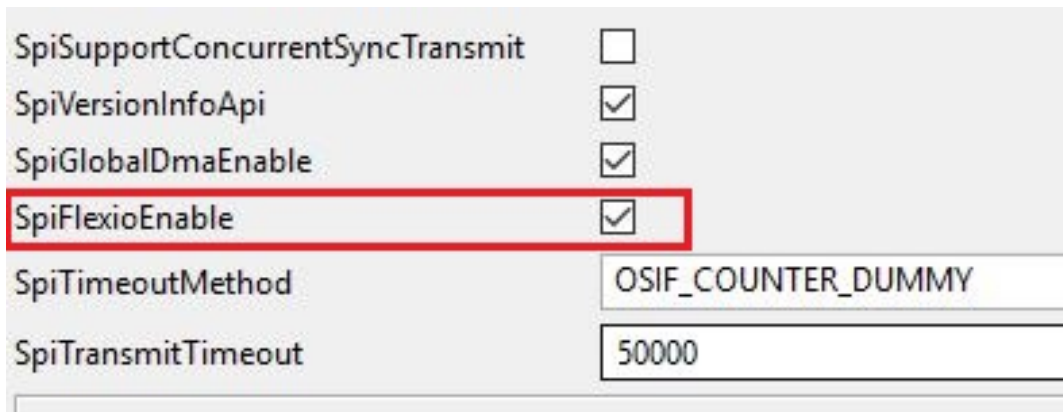


Figure 3.21 Enable SPI over FLEXIO

- Enable Flexio support on MCL site: (bellow picture)



Figure 3.22 Enable Mcl FLEXIO support

- In MCL, setup a set for SPI FLEXO Channels like: (bellow picture)
- Flexio Channel

is no. of Shifter and Timer registers will be selected for each FLEXIO channel. For example: CHANNEL_0 selected means Shifter_0 and Timer_0 will be selected and locked, other channels cannot select those registers.

- Flexio Pin

is no. of PIN will be selected for each FLEXIO channel. For example: PIN_0 selected means FXIO_D0 will be selected and locked, other channels cannot select this PIN.

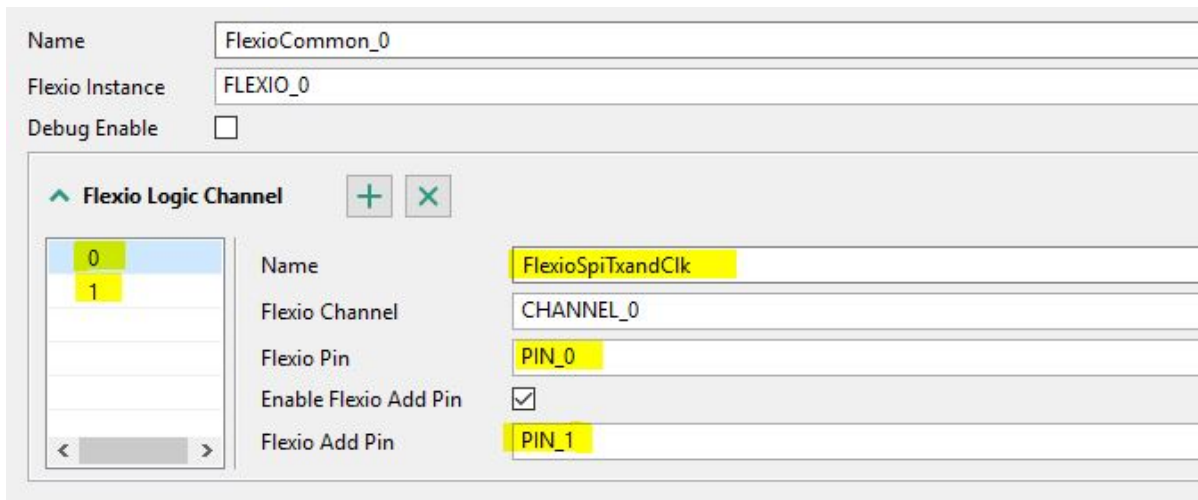


Figure 3.23 A set of FLEXIO SPI channels from MCL

- From SPI, each FLEXIO_SPI_x is selected in SpiPhyUnit then those set must be enabled and selected correctly value (refer from MCL which was configured above) (bellow picture)
 - SpiFlexioTxAndClkChannelsConfig: FLEXIO Logical Channel for TX(MOSI) and CLK(Clock). This selects a PIN to setup as TX(MOSI) channel and another for CLK channel. If you select CHANNEL_X, PIN_Y(FlexioMclPinId) and PIN_Z(FlexioMclAddPinId) that means:
 - * ShifterX is selected for TX(MOSI) channel
 - * TimerX is selected for CLK channel.
 - * PIN_Y is selected for TX(MOSI) channel
 - * PIN_Z is selected for CLK channel
 - SpiFlexioRxAndCsChannelsConfig: FLEXIO Logical Channel for RX(MISO) and CS(Chip select). This selects 1 PIN to setup as MISO channel and 1 PIN to set up as CS channel. If you select CHANNEL_X, PIN_Y(FlexioMclPinId) and PIN_Z(FlexioMclAddPinId) that mean:
 - * ShifterX is selected for RX(MISO) channel
 - * TimerX is selected for CS channel.
 - * PIN_Y is selected for RX(MISO) channel
 - * PIN_Z is selected for CS channel

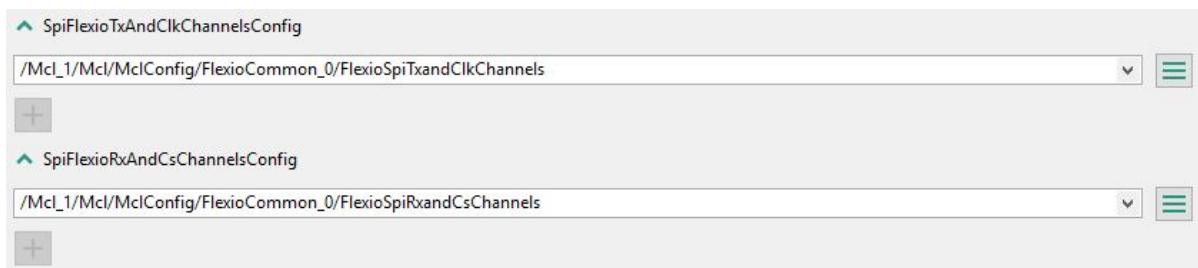


Figure 3.24 A set of FLEXIO SPI channels from SPI

3.6.5.3 Register ISR for INTERRUPT mode

- ISR name: **MCL_FLEXIO_ISR**

- Register FLEXIO ISR like document. For example:

```
sys_disableIsrSource(69);
sys_enableIsrSource(69, 7);
sys_registerIsrHandler(69, MCL_FLEXIO_ISR);
```

where:

139: NVIC Interrupt ID of FLEXIO module

7: The priority

3.6.6 How to configure to use half duplex mode

To configure Spi in half duplex mode that has some different to normal configuration of full duplex mode. The different will proposed below:

- First of all, the SpiHalfDuplexModeSupport checkbox in SpiAutosarExt must be enabled to half-duplex mode can be configured in configurational tools:



Figure 3.25 Enable Half-duplex mode support on EBTresos

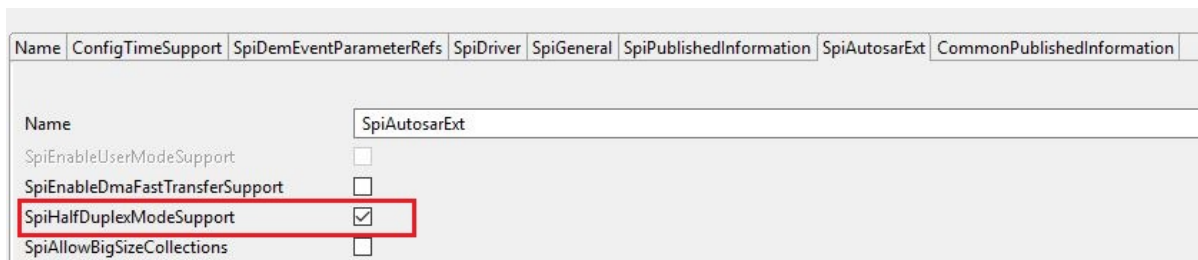


Figure 3.26 Enable Half-duplex mode support on S32 Design Studio

- Second, an external device has to be created with half-duplex transfer support by enabling SpiDeviceHalfDuplexSupport checkbox. After that, selecting number of bit transmitted and received on each clock edge. There are 3 types of transfer are supported: 1-bit, 2-bits or 4 bits.

Note

- Half-duplex mode only supports in external devices which are using Lpspi.
- With 1-bit transfer, the CIN or COUT pin could be chosen by SpiHalfDuplexPinSelect on External↔Device configuration.
- With 2-bits transfer, the CIN and COUT pins will be used.
- And with 4-bits transfer, the CIN, COUT, PCS2 and PCS3 will be used.

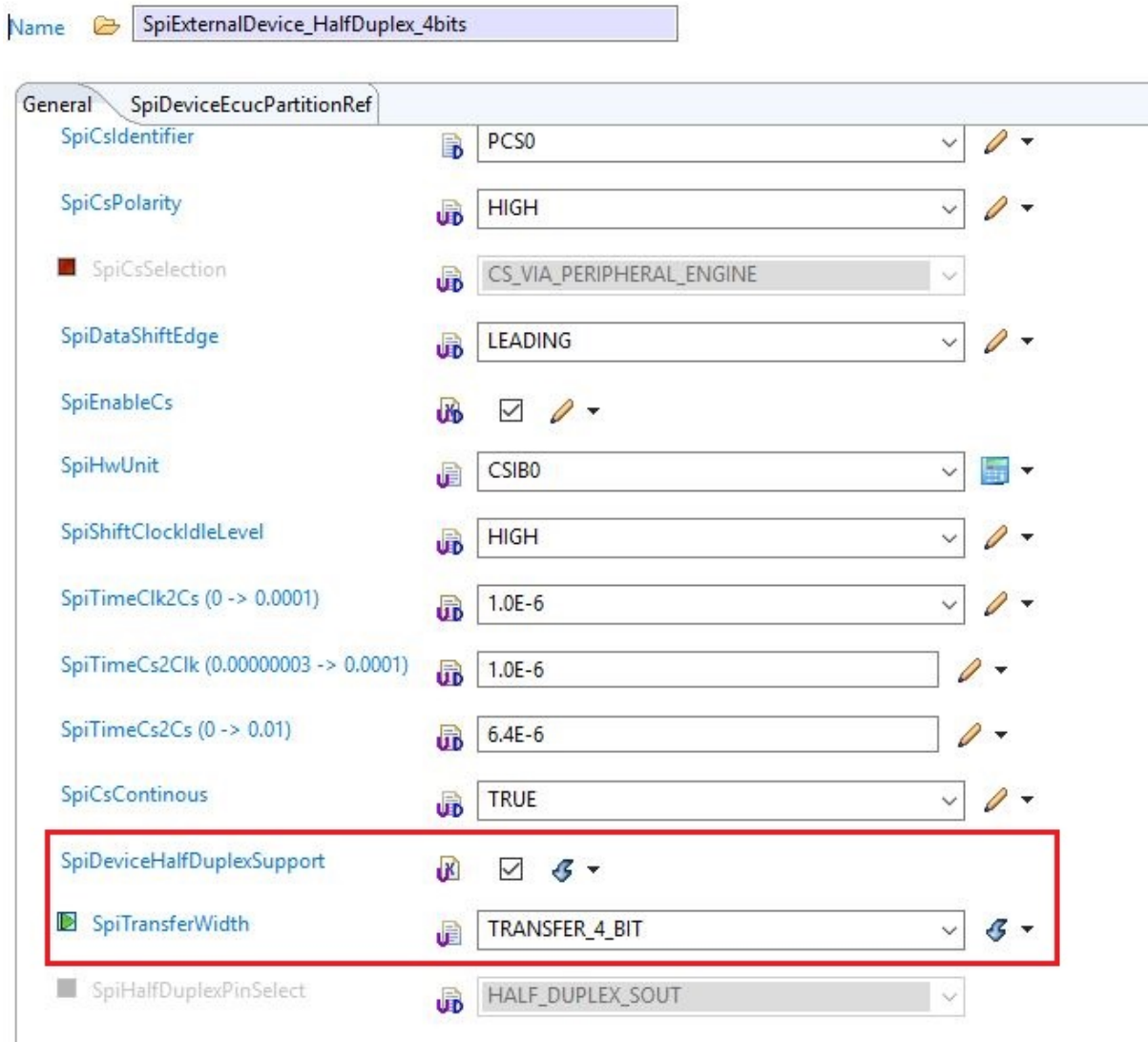


Figure 3.27 Configure Half-duplex mode for external device on EBTresos



Figure 3.28 Configure Half-duplex mode for external device on S32 Design Studio

- Finish, a channel should be selected as transmission or reception in half-duplex mode by configuring the Spi↔ChannelHalfDuplexDirection.

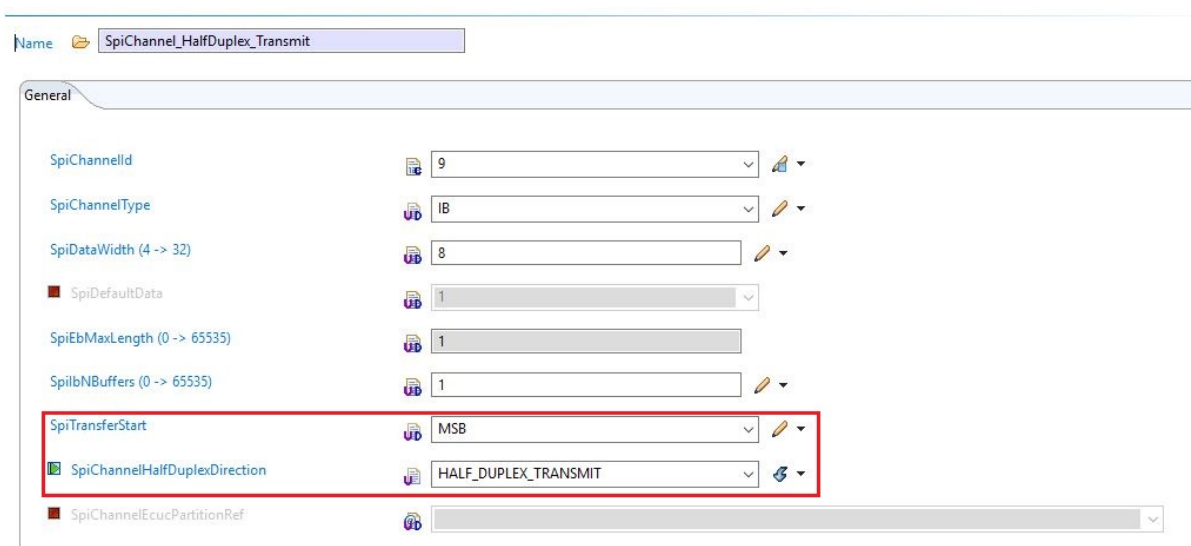


Figure 3.29 Configure Half-duplex mode for external device on EBTresos

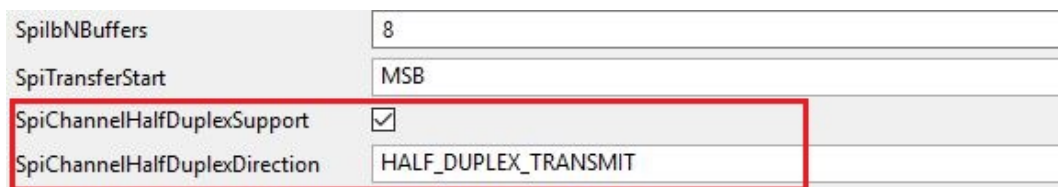


Figure 3.30 Configure Half-duplex mode for external device on S32 Design Studio

3.6.7 How to use Half Duplex mode

Below is presenting on EB tresos, this is same on CT.

- Enable Spi half duplex mode support:

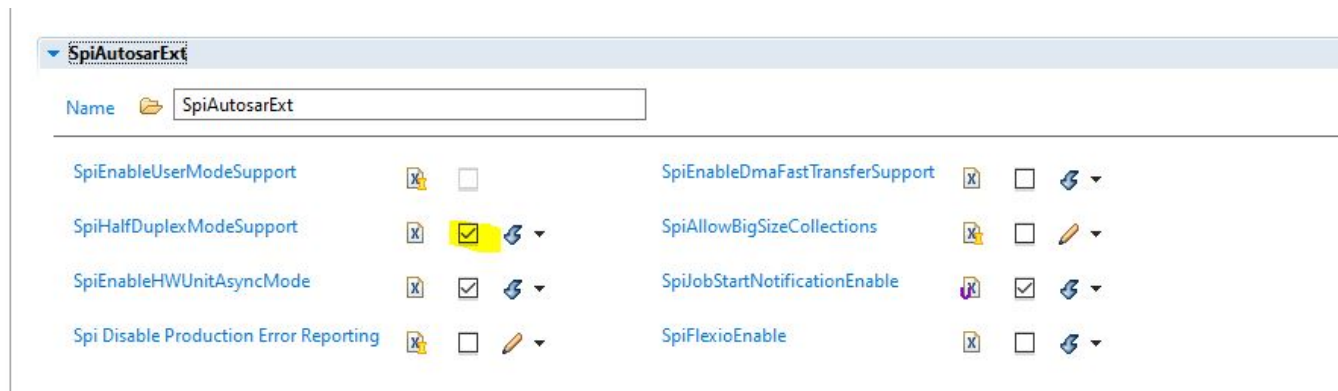



Figure 3.31 Enable Spi half duplex mode support

- Select direction for each channel:
 - HALF_DUPLEX_TRANSMIT: Transmit only mode
 - HALF_DUPLEX_RECEIVE: Receive only mode

SpiChannel

Name  SpiChannel_0

General



















SpiChannelId	 0	
SpiChannelType	 EB	
SpiDataWidth (4 -> 32)	 8	
 SpiDefaultData	 1	
SpiEbMaxLength (0 -> 65535)	 10	
SpiNbBuffers (0 -> 65535)	 1	
SpiTransferStart	 LSB	
 SpiChannelHalfDuplexDirection	 HALF_DUPLEX_TRANSMIT	
 SpiChannelEcucPartitionRef		

Figure 3.32 Direction of channels

- Configure Externaldevice:
 - SpiTransferWidth: Number of bits will be transfered on per cycle clock. It supports 1/2/4 bits on Master mode, Slave mode only supports 1 bit mode.
 - SpiHalfDuplexPinSelect: Pin will be selected for the transfer (it is HALF_DUPLEX_SOUT means MOSI will be selected).

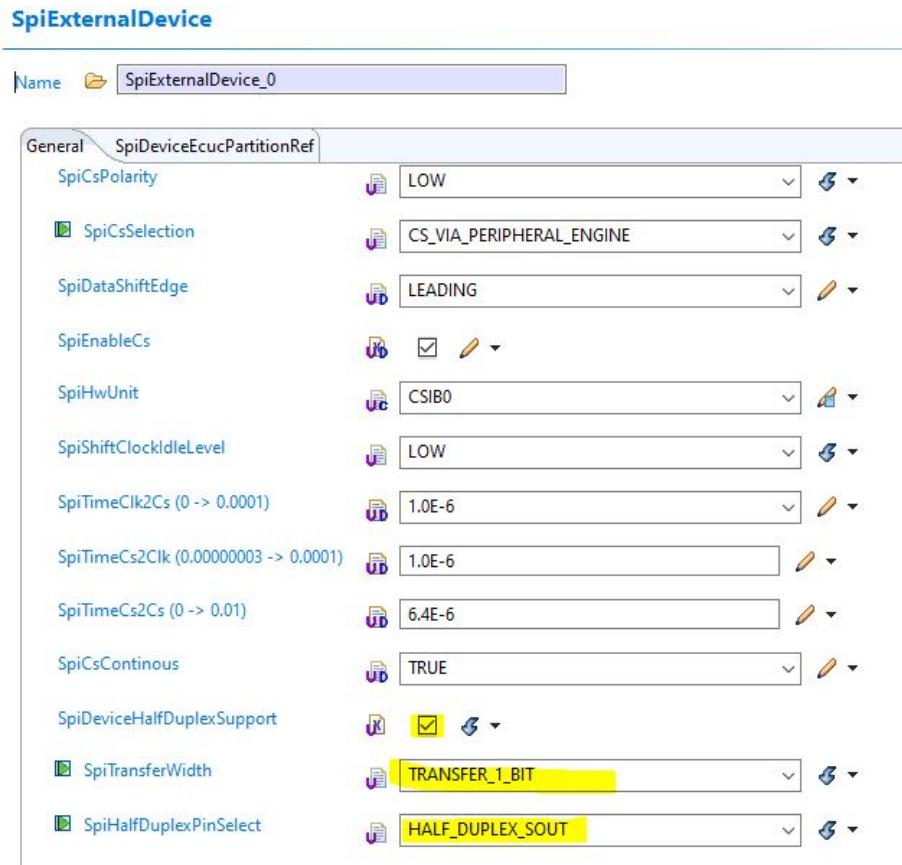


Figure 3.33 Direction of channels

3.6.8 Pointer to buffer alignment

- For $9 \leq \text{Framesize} < 17$: User or integrator have to make sure that pointer to 2-byte aligned buffer.
- For $17 \leq \text{Framesize}$: User or integrator have to make sure that pointer to 2-byte aligned buffer.

3.7 Runtime errors

The driver generates the following DEM errors at runtime.

Function	Error Code	Condition triggering the error
Spi_SyncTransmit	Spi_E_Hardware_ErrorCfg	The SPI driver cannot transmit complete or receive complete one frame in the allocated time defined by “SP↔I_TIMEOUT_COUNTER” parameter in configuration. Baud rate of HW might be low speed or timeout value too short. Timeout occurred.
Spi_SyncTransmit	SPI_E_SEQ_IN_PROCESS	Synchronous transmission service called at wrong time.
Spi_AsyncTransmit	SPI_E_SEQ_PENDING	Services called in a wrong sequence.

3.8 Symbolic Names Disclaimer

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

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

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

Chapter 4

Tresos Configuration Plug-in

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

- Module [Spi](#)
 - Container [SpiDemEventParameterRefs](#)
 - * Reference [SPI_E_HARDWARE_ERROR](#)
 - Container [SpiDriver](#)
 - * Parameter [SpiMaxChannel](#)
 - * Parameter [SpiMaxJob](#)
 - * Parameter [SpiMaxSequence](#)
 - * Container [SpiChannel](#)
 - Parameter [SpiChannelId](#)
 - Parameter [SpiChannelType](#)
 - Parameter [SpiDataWidth](#)
 - Parameter [SpiDefaultData](#)
 - Parameter [SpiEbMaxLength](#)
 - Parameter [SpiIbNBuffers](#)
 - Parameter [SpiTransferStart](#)
 - Parameter [SpiChannelHalfDuplexDirection](#)
 - Reference [SpiChannelEcucPartitionRef](#)
 - * Container [SpiExternalDevice](#)
 - Parameter [SpiBaudrate](#)
 - Parameter [SpiCsIdentifier](#)
 - Parameter [SpiCsPolarity](#)
 - Parameter [SpiCsSelection](#)
 - Parameter [SpiDataShiftEdge](#)
 - Parameter [SpiEnableCs](#)
 - Parameter [SpiHwUnit](#)
 - Parameter [SpiShiftClockIdleLevel](#)
 - Parameter [SpiTimeClk2Cs](#)
 - Parameter [SpiTimeCs2Clk](#)
 - Parameter [SpiTimeCs2Cs](#)
 - Parameter [SpiCsBehavior](#)

- Parameter [SpiDeviceHalfDuplexSupport](#)
- Parameter [SpiTransferWidth](#)
- Parameter [SpiHalfDuplexPinSelect](#)
- Reference [SpiDeviceEcucPartitionRef](#)
- * Container [SpiJob](#)
 - Parameter [SpiJobEndNotification](#)
 - Parameter [SpiJobStartNotification](#)
 - Parameter [SpiJobId](#)
 - Parameter [SpiJobPriority](#)
 - Reference [SpiDeviceAssignment](#)
 - Container [SpiChannelList](#)
 - Parameter [SpiChannelIndex](#)
 - Reference [SpiChannelAssignment](#)
- * Container [SpiSequence](#)
 - Parameter [SpiInterruptibleSequence](#)
 - Parameter [SpiSeqEndNotification](#)
 - Parameter [SpiSequenceId](#)
 - Parameter [SpiEnableDmaFastTransfer](#)
 - Reference [SpiJobAssignment](#)
- Container [SpiGeneral](#)
 - * Parameter [SpiMulticoreSupport](#)
 - * Parameter [SpiCancelApi](#)
 - * Parameter [SpiChannelBuffersAllowed](#)
 - * Parameter [SpiDevErrorDetect](#)
 - * Parameter [SpiHwStatusApi](#)
 - * Parameter [SpiInterruptibleSeqAllowed](#)
 - * Parameter [SpiLevelDelivered](#)
 - * Parameter [SpiMainFunctionPeriod](#)
 - * Parameter [SpiSupportConcurrentSyncTransmit](#)
 - * Parameter [SpiVersionInfoApi](#)
 - * Parameter [SpiGlobalDmaEnable](#)
 - * Parameter [SpiTimeoutMethod](#)
 - * Parameter [SpiTransmitTimeout](#)
 - * Reference [SpiEcucPartitionRef](#)
 - * Reference [SpiKernelEcucPartitionRef](#)
 - * Container [SpiPhyUnit](#)
 - Parameter [SpiPhyUnitMapping](#)
 - Parameter [SpiPhyUnitMode](#)
 - Parameter [SpiPhyUnitSync](#)
 - Parameter [SpiSamplePoint](#)
 - Parameter [SpiPinConfiguration](#)
 - Parameter [SpiPhyUnitAsyncUseDma](#)
 - Parameter [SpiMaxDmaFastTransfer](#)
 - Reference [SpiPhyUnitClockRef](#)

- Reference [SpiPhyUnitAlternateClockRef](#)
- Reference [SpiPhyTxDmaChannel](#)
- Reference [SpiPhyRxDmaChannel](#)
- Reference [SpiFlexioTxAndClkChannelsConfig](#)
- Reference [SpiFlexioRxAndCsChannelsConfig](#)
- Container [SpiPublishedInformation](#)
 - * Parameter [SpiMaxHwUnit](#)
- Container [CommonPublishedInformation](#)
 - * Parameter [ArReleaseMajorVersion](#)
 - * Parameter [ArReleaseMinorVersion](#)
 - * Parameter [ArReleaseRevisionVersion](#)
 - * Parameter [ModuleId](#)
 - * Parameter [SwMajorVersion](#)
 - * Parameter [SwMinorVersion](#)
 - * Parameter [SwPatchVersion](#)
 - * Parameter [VendorApiInfix](#)
 - * Parameter [VendorId](#)
- Container [SpiAutosarExt](#)
 - * Parameter [SpiEnableUserModeSupport](#)
 - * Parameter [SpiEnableDmaFastTransferSupport](#)
 - * Parameter [SpiHalfDuplexModeSupport](#)
 - * Parameter [SpiAllowBigSizeCollections](#)
 - * Parameter [SpiEnableHWUnitAsyncMode](#)
 - * Parameter [SpiJobStartNotificationEnable](#)
 - * Parameter [SpiDisableDemReportErrorStatus](#)
 - * Parameter [SpiFlexioEnable](#)

4.1 Module Spi

Configuration of the Spi (Serial Peripheral Interface) module.

Included containers:

- [SpiDemEventParameterRefs](#)
- [SpiDriver](#)
- [SpiGeneral](#)
- [SpiPublishedInformation](#)
- [CommonPublishedInformation](#)
- [SpiAutosarExt](#)

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

4.2 Container SpiDemEventParameterRefs

Container for the references to DemEventParameter elements which shall be invoked using the Dem_SetEventStatus 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 subcontainers:

- None

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

4.3 Reference SPI_E_HARDWARE_ERROR

Reference to configured DEM event to report "Hardware failure". If the reference is not configured the error shall not be reported.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE

Property	Value
	VARIANT-POST-BUILD: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/Dem/DemConfigSet/DemEventParameter

4.4 Container SpiDriver

This container contains the configuration parameters and sub containers of the AUTOSAR Spi module.

Included subcontainers:

- [SpiChannel](#)
- [SpiExternalDevice](#)
- [SpiJob](#)
- [SpiSequence](#)

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

4.5 Parameter SpiMaxChannel

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

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	0
max	65535
min	0

4.6 Parameter SpiMaxJob

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

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	0
max	65535
min	0

4.7 Parameter SpiMaxSequence

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

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	0
max	65535
min	0

4.8 Container SpiChannel

All data needed to configure one SPI-channel.

Included subcontainers:

- None

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

4.9 Parameter SpiChannelId

SPI Channel ID, used as parameter in SPI API functions.

Property	Value
type	ECUC-INTEGER-PARAM-DEF

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

4.10 Parameter SpiChannelType

Buffer usage with EB/IB channel.

Note 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

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

4.11 Parameter SpiDataWidth

This parameter is the width of a transmitted data unit.

NoteThe hardware supports data width from 1 to 64 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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	32
max	64
min	1

4.12 Parameter SpiDefaultData

The default data to be transmitted when (for internal buffer or external buffer)

the pointer passed to Spi_WriteIB (for internal buffer) or to Spi_SetupEB (for external buffer) is NULL.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD

Property	Value
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	1
max	4294967295
min	0

4.13 Parameter SpiEbMaxLength

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	1024
max	1048576
min	1

4.14 Parameter SpiIbNBuffers

This parameter contains the maximum number of data buffers in case of IB Channels and only.

In case of channel's Spi_DataWidth ranges from 9 to 16, this parameter refers to

the number of bytes allocated to the buffers and MUST be even. Or divisible by 4 if the range from 17 to 32.

Property	Value
type	ECUC-INTEGER-PARAM-DEF

Property	Value
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	1
max	65535
min	1

4.15 Parameter SpiTransferStart

This parameter defines the first starting bit for transmission.

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

4.16 Parameter SpiChannelHalfDuplexDirection

This parameter select direction of transfer in half duplex mode

HALF_DUPLEX_TRANSMIT: Transmit only.

HALF_DUPLEX_RECEIVE: Receive only, supports DataWidth 1->32bits.

Note This parameter only is used in Half Duplex mode.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	HALF_DUPLEX_TRANSMIT
literals	['HALF_DUPLEX_TRANSMIT', 'HALF_DUPLEX_RECEIVE']

4.17 Reference SpiChannelEcucPartitionRef

Maps an SPI Channel to zero or ECUC partition to limit the access to this Channel. The ECUC partition referenced is a

subset of the ECUC partitions where the SPI driver is mapped to.

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

4.18 Container SpiExternalDevice

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

Included subcontainers:

- None

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

4.19 Parameter SpiBaudrate

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.

This field is used only for LPSPI

The baudrate must be in allowed range

LPSPI in normal Master mode:

- RUN MODE: [1 to 10Mhz]
- HSRUN MODE: [1 to 14Mhz] only support for K14x
- VLPR MODE: [1 to 2Mhz] for K1xx and [1 to 500Khz] for K14xW

FLEXIO in Master:

Note: Due to synchronization delays, the setup time for the serial input data is 1.5 FlexIO clock cycles, so the maximum baud rate is divide by 4 of the FlexIO clock frequency.

Theoretical baud rate =

$$\text{FLEXIO_CLK (BUS_CLK)} \div (16 \div 2 \div (\text{TIMCMPn[COMP]} + 1))$$

$$\text{FLEXIO_CLK (BUS_CLK)} \div (2 \div (\text{TIMCMPn[COMP]} + 1))$$

- RUN MODE: [1 to $(\text{BUS_CLK}/4) \div (2 \div (\text{TIMCMPn[COMP]} + 1)) = 6\text{Mhz}$]
- HSRUN MODE: [1 to $(\text{BUS_CLK}/4) \div (2 \div (\text{TIMCMPn[COMP]} + 1)) = 7\text{Mhz}$] only support for K14x
- VLPR MODE: [1 to $(\text{BUS_CLK}/4) \div (2 \div (\text{TIMCMPn[COMP]} + 1)) = 0.5\text{Mhz}$] for K11x, K14x and [1 to $(\text{BUS_CLK}/4) \div (2 \div (\text{TIMCMPn[COMP]} + 1)) = 62.5\text{KhzMhz}$] for K14xW

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.

Property	Value
type	ECUC-FLOAT-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	1000000.0
max	1.4E7
min	1.0

4.20 Parameter SpiCsIdentifier

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

The chip selects are specific per HwUnit. Please check in Reference Manual for information on available chip selects.

If FLEXIO channel used, Chip Select will be configured by SpiPhyUnit\SpiFlexioCsPinSelect and this parameter will be not used.

If SpiEnableCs is not set, value of this node will not be used by driver code. It should set to default value (PCS0)

Property	Value
type	ECUC-STRING-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	true
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	PCS0

4.21 Parameter SpiCsPolarity

This parameter defines the active polarity of Chip Select.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
default Value	HIGH
literals	['HIGH', 'LOW']

4.22 Parameter SpiCsSelection

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.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
default Value	CS_VIA_PERIPHERAL_ENGINE
literals	['CS_VIA_PERIPHERAL_ENGINE', 'CS_VIA_GPIO']

4.23 Parameter SpiDataShiftEdge

This parameter defines the SPI data shift edge.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
default Value	LEADING
literals	['LEADING', 'TRAILING']

4.24 Parameter SpiEnableCs

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.

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

4.25 Parameter SpiHwUnit

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 SPI_0 HW unit).

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	CSIB0
literals	['CSIB0', 'CSIB1', 'CSIB2', 'CSIB3', 'CSIB4']

4.26 Parameter SpiShiftClockIdleLevel

This parameter defines the SPI shift clock idle level.

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

4.27 Parameter SpiTimeClk2Cs

Timing between clock and chip select in seconds (tASC) - This parameter allows to use a range of values

from 0.00000001 up to 0.01 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.

Property	Value
type	ECUC-FLOAT-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	1.0E-6
max	0.01
min	1.0E-8

4.28 Parameter SpiTimeCs2Clk

Timing between chip select and clock in seconds (tCSC) - This parameter allows to use a range of values from 0.00000003 up to 0.01 Sec.

If use continuous transfer(PCS signals remain asserted between transfers), tASC and tCSC will insert between transfers.

NoteThis is an implementation specific parameter.

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

Property	Value
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	1.0E-6
max	0.01
min	1.0E-8

4.29 Parameter SpiTimeCs2Cs

Timing between chip select assertions in seconds (tDT) - This parameter allows to use a range of values from 0.00000001 up to 0.01 Sec.

If use continuous transfer(PCS signals remain asserted between transfers), tDT is not inserted between the transfers.

Note This is an implementation parameter.

Property	Value
type	ECUC-FLOAT-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	1.0E-6
max	0.01
min	1.0E-8

4.30 Parameter SpiCsBehavior

This parameter is used to define the chip select behavior.

Either the CS is toggled for each data frame (bit frame on the SPI bus in relation with SpiDataWidth) inside the channel(s)

composing the job or the CS is kept asserted for the whole job.

Note: For Flexio, don't use these configurations: (SPI_SLAVE, TRAILING), (SPI_SLAVE, LEADING, SpiShift-ClockIdleLevel = HIGH).

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

4.31 Parameter SpiDeviceHalfDuplexSupport

This parameter enables or not half duplex mode for this external device.

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

4.32 Parameter SpiTransferWidth

In half duplex mode, this will select the number of bits are transfered on per cycle clock

TRANSFER_1_BIT: 1 bit will be transfered on per cycle clock

TRANSFER_2_BIT: 2 bits will be transfered on per cycle clock

TRANSFER_4_BIT: 4 bits will be transfered on per cycle clock

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	TRANSFER_1_BIT
literals	['TRANSFER_1_BIT', 'TRANSFER_2_BIT', 'TRANSFER_4_BIT']

4.33 Parameter SpiHalfDuplexPinSelect

This node will support to select which Pin (SIN or SOUT) will be used for half duplex 1 bit mode.

HALF_DUPLEX_SIN: SIN will be used to transfer

HALF_DUPLEX_SOUT: SOUT will be used to transfer

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	HALF_DUPLEX_SOUT
literals	['HALF_DUPLEX_SIN', 'HALF_DUPLEX_SOUT']

4.34 Reference SpiDeviceEcucPartitionRef

ECUC_Spi_00246. Maps an SPI external device to zero or multiple ECUC partitions to limit the access to this external device. The ECUC partitions referenced are a subset of the ECUC partitions where the SPI driver is mapped to.

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

4.35 Container 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.

Included subcontainers:

- [SpiChannelList](#)

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

4.36 Parameter SpiJobEndNotification

This parameter is a reference to a notification function.

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

4.37 Parameter SpiJobStartNotification

This parameter is a reference to a notification function.

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

4.38 Parameter SpiJobId

SPI Job ID, used as parameter in SPI API functions.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	true
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
default Value	0
max	65535
min	0

4.39 Parameter SpiJobPriority

Priority set accordingly to SPI093: 0, lowest, 3, highest priority

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

4.40 Reference SpiDeviceAssignment

Reference to the external device used by this job.

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

4.41 Container SpiChannelList

References to SPI channels and their order within the Job.

Included subcontainers:

- None

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

4.42 Parameter SpiChannelIndex

This parameter specifies the order of Channels within the Job.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	0
max	65535
min	0

4.43 Reference SpiChannelAssignment

A job references several channels.

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

4.44 Container SpiSequence

All data needed to configure one SPI-sequence.

Included subcontainers:

- None

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

4.45 Parameter SpiInterruptibleSequence

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

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

4.46 Parameter SpiSeqEndNotification

This parameter is a reference to a notification function.

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

Property	Value
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
default Value	NULL_PTR

4.47 Parameter SpiSequenceId

Sequence ID of configured SPI Sequence.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	true
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
default Value	0
max	255
min	0

4.48 Parameter SpiEnableDmaFastTransfer

When this parameter is enabled, this Sequence will be transferred using DMA ScatterGather and CPU used only for processing end of Sequence.

SpiAutosarExt/SpiEnableDmaFastTransferSupport must be checked to support this feature.

Note: This feature requires:

1. The parameters SpiBaudrate, SpiHwUnit, SpiTimeClk2Cs, SpiTimeCs2Clk, SpiTimeCs2Cs in External Device linked to each Job in this Sequence must be the same.
2. The parameters SpiDataWidth and SpiTransferStart in Channel assigned to each Job in this Sequence must be the same.
3. In each Channel, the number of data buffers is NOT higher than 32767 if SpiDataWidth < 9. So, SpiIbNBuffers and SpiEbMaxLength must be assigned to suitable values.
4. Only Master mode is supported.

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

4.49 Reference SpiJobAssignment

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

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	1
upperMultiplicity	Infinite
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/Spi/SpiDriver/SpiJob

4.50 Container SpiGeneral

General configuration settings for SPI-Handler.

Included subcontainers:

- [SpiPhyUnit](#)

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

4.51 Parameter SpiMulticoreSupport

SpiMulticoreEnable

When this parameter is enabled, multi-core feature will be used in SPI driver.

That means mapping the SPI driver to multiple ECUC partitions to make the module API available in this partition.

The SPI driver will operate as an independent instance in each of the partitions.

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

4.52 Parameter SpiCancelApi

Switches the Spi_Cancel function ON or OFF.

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

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

4.53 Parameter SpiChannelBuffersAllowed

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

Note

0 - Only Internal Buffers (IB) are allowed

1 - Only External buffers (EB) are allowed

2 - Both Internal (IB) and External (EB) buffers are allowed

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

4.54 Parameter SpiDevErrorDetect

Switches the Development Error Detection and Notification ON or OFF.

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

4.55 Parameter SpiHwStatusApi

Switches the Spi_GetHWUnitStatus function ON or OFF.

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

4.56 Parameter SpiInterruptibleSeqAllowed

Switches the Interruptible Sequences handling functionality ON or OFF.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false

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

4.57 Parameter SpiLevelDelivered

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

Note

Level 0 Only Simple Synchronous Behavior

Level 1 Basic Asynchronous Behaviour

Level 2 Enhanced Behaviour

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
default Value	2
max	2
min	0

4.58 Parameter SpiMainFunctionPeriod

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.

Property	Value
type	ECUC-FLOAT-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	0.01
max	1.0
min	1.0E-7

4.59 Parameter SpiSupportConcurrentSyncTransmit

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

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

4.60 Parameter SpiVersionInfoApi

Switches the Spi_GetVersionInfo function ON or OFF.

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

4.61 Parameter SpiGlobalDmaEnable

If checked, it allows using the DMA module during the transfer.

For each SPI unit a transferring method can be configured: FIFO or DMA.

If not checked, all SPI units will use FIFO transferring mode.

Note This is an implementation parameter.

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

4.62 Parameter SpiTimeoutMethod

SpiTimeoutMethod

Tresos Configuration Plug-in

Configures the timeout method.

Based on this selection a certain timeout method from OsIf will be used in the driver.

Note: If SystemTimer or CustomTimer are selected make sure the corresponding timer is enabled in OsIf General configuration.

Note: Implementation Specific Parameter.

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

4.63 Parameter SpiTransmitTimeout

Timeout value (microseconds) used to wait for TX/RX transmission to complete one frame in both full duplex and half duplex mode NoteThis is an implementation parameter. The transmission will be unsuccessful if the Chip cannot completely transfer one frame during this timeout.

The precision of this value is quite low, it must be greater than the time needed to completely transmit one frame.

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

Property	Value
defaultValue	50000
max	900000
min	1

4.64 Reference SpiEcucPartitionRef

ECUC_Spi_00244.Maps the SPI driver to zero or multiple ECUC partitions to make the driver

API available in the according partition.

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

4.65 Reference SpiKernelEcucPartitionRef

ECUC_Spi_00245.Maps the SPI kernel to zero or one ECUC partitions to assign the driver

kernel to a certain core. The ECUC partition referenced is a subset of the

ECUC partitions where the SPI driver is mapped to. SPI driver is implemented according to multicore type II, so this node is not used.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true

Property	Value
multiplicityConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcuDefs/EcuC/EcuPartitionCollection/EcuPartition

4.66 Container SpiPhyUnit

Logical to Physical SPI Bus mapping.

Note This is an implementation specific container.

Included subcontainers:

- None

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

4.67 Parameter SpiPhyUnitMapping

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

Note This is an implementation specific parameter.

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

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	LPSPI_1
literals	['LPSPI_0', 'LPSPI_1', 'LPSPI_2', 'FLEXIO_SPI_0', 'FLEXIO_SPI_1']

4.68 Parameter SpiPhyUnitMode

Select between SPI_MASTER and SPI_SLAVE modes.

SPI Slave mode support only if SpiGeneral/SpiLevelDelivered is 1 or 2

Note This is an implementation specific parameter.

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

4.69 Parameter SpiPhyUnitSync

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.

Note This is an implementation specific parameter.

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

4.70 Parameter SpiSamplePoint

When set, the LPSPI master will sample the input data on a delayed LPSPI_SCK edge, which improves the setup time when sampling data.

? The input data setup time in master mode with delayed LPSPI_SCK edge is equal to the input data setup time in slave mode

? In slave mode, the SAMPLE bit is ignored

0b - Input data is sampled on SCK edge

1b - Input data is sampled on delayed SCK edge

Note This is an implementation specific parameter.

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

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

4.71 Parameter SpiPinConfiguration

Configures which pins are used for input and output data during serial transfers. When performing parallel transfers, the Pin Configuration field is ignored.

00b - SIN is used for input data and SOUT is used for output data

01b - SIN is used for both input and output data, only half-duplex serial transfers are supported

10b - SOUT is used for both input and output data, only half-duplex serial transfers are supported

11b - SOUT is used for input data and SIN is used for output data

Note This is an implementation specific parameter.

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

4.72 Parameter SpiPhyUnitAsyncUseDma

Select Asynchronous mechanism with DMA or not.

Note This is an implementation parameter.

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

4.73 Parameter SpiMaxDmaFastTransfer

Number of transfer section allowed in Dma Fast transfer.

Note: This feature will be supported if SpiEnableDmaFastTransferSupport and SpiPhyUnitAsyncUseDma checked.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
default Value	1
max	10
min	1

4.74 Reference SpiPhyUnitClockRef

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

This clock source is used for configure SPI baudrate.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/Mcu/McuModuleConfiguration/McuClockSetting↔ Config/McuClockReferencePoint

4.75 Reference SpiPhyUnitAlternateClockRef

Reference to the alternate clock configuration, retrieved from the MCU plugin.

Use to enable Spi_SetClockMode() function, which allows dual MCU clock configuration settings.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/Mcu/McuModuleConfiguration/McuClockSetting↔ Config/McuClockReferencePoint

4.76 Reference SpiPhyTxDmaChannel

SPI Master Transmit DMA Logical Channel as configured by MCL plug-in, used to prepare the SPI transmission dataframes

starting from the TX buffer content.

This parameter is required only if SpiPhyUnitAsyncUseDma is checked.

Note This is an implementation specific parameter. The current SPI TX source

needs be configured for enabling this DMA channel.

Property	Value
type	ECUC-CHOICE-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
requiresSymbolicNameValue	False
destinations	['/TS_T40D2M20I0R0/Mcl/MclConfig/dmaLogicChannel_Type']

4.77 Reference SpiPhyRxDmaChannel

SPI 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 SpiPhyUnitAsyncUseDma is checked.

Note This is an implementation specific parameter.

Property	Value
type	ECUC-CHOICE-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE

Property	Value
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
postBuildVariantValue	false
	VARIANT-LINK-TIME: PRE-COMPILE
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
requiresSymbolicNameValue	False
destinations	[! /TS_T40D2M20I0R0/Mcl/MclConfig/dmaLogicChannel_Type]

4.78 Reference SpiFlexioTxAndClkChannelsConfig

FLEXIO Logical Channel for TX(MOSI) and CLK(Clock). This selects 1 PIN to setup as TX(MOSI) channel and 1 PIN to set up as CLK channel

If you select CHANNEL_X, PIN_Y(FlexioMclPinId) and PIN_Z(FlexioMclAddPinId) that mean:

- ShifterX is selected for TX(MOSI) channel
- TimerX is selected for CLK channel.
- PIN_Y is selected for TX(MOSI) channel
- PIN_Z is selected for CLK channel

Note This is an implementation specific parameter.

Property	Value
type	ECUC-CHOICE-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
	VARIANT-LINK-TIME: PRE-COMPILE
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
postBuildVariantValue	false
	VARIANT-LINK-TIME: PRE-COMPILE
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
requiresSymbolicNameValue	False
destinations	[! /TS_T40D2M20I0R0/Mcl/MclConfig/FlexioCommon/FlexioMclLogicChannels]

4.79 Reference SpiFlexioRxAndCsChannelsConfig

FLEXIO Logical Channel for RX(MISO) and CS(Chip select). This selects 1 PIN to setup as MISO channel and 1 PIN to set up as CS channel

If you select CHANNEL_X, PIN_Y(FlexioMclPinId) and PIN_Z(FlexioMclAddPinId) that mean:

- ShifterX is selected for RX(MISO) channel
- TimerX is selected for CS channel.
- PIN_Y is selected for RX(MISO) channel
- PIN_Z is selected for CS channel

Note This is an implementation specific parameter.

Property	Value
type	ECUC-CHOICE-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
requiresSymbolicNameValue	False
destinations	['/TS_T40D2M20I0R0/Mcl/MclConfig/FlexioCommon/FlexioMclLogic↵ Channels', '/TS_T40D2M20I0R0/Mcl/MclConfig/FlexioCommon/Flexio↵ MclLogicChannels']

4.80 Container SpiPublishedInformation

Container holding all SPI specific published information parameters.

Included subcontainers:

- None

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

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

4.81 Parameter SpiMaxHwUnit

Number of different SPI hardware microcontroller peripherals (units/busses) available and handled by this SPI Handler/Driver module.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PUBLISHED-INFORMATION
	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	65535
min	0

4.82 Container CommonPublishedInformation

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

Included subcontainers:

- None

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

4.83 Parameter ArReleaseMajorVersion

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

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

4.84 Parameter ArReleaseMinorVersion

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	7
max	7
min	7

4.85 Parameter ArReleaseRevisionVersion

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

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

4.86 Parameter ModuleId

Module ID of this module.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	83
max	83
min	83

4.87 Parameter SwMajorVersion

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

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

4.88 Parameter SwMinorVersion

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

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

4.89 Parameter SwPatchVersion

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

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

4.90 Parameter VendorApiInfix

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

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

<ModuleName>__>VendorId>__<VendorApiInfix>.

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

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

Property	Value
type	ECUC-STRING-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-POST-BUILD: POST-BUILD

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

4.91 Parameter VendorId

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

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

4.92 Container SpiAutosarExt

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

Included subcontainers:

- None

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

4.93 Parameter SpiEnableUserModeSupport

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

Note Spi module does not include registers protection. So, It is accessible to all registers in any public mode.

SPI is not affected by this field.

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

4.94 Parameter SpiEnableDmaFastTransferSupport

When this parameter is enabled, the SPI module can support to transfer a Sequence with multiple Channels, Jobs using DMA ScatterGather and CPU used only for processing end of sequence transfer.

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

4.95 Parameter SpiHalfDuplexModeSupport

When this parameter is enabled, the SPI module can support to transfer in half duplex mode.

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

4.96 Parameter SpiAllowBigSizeCollections

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

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

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

4.97 Parameter SpiEnableHWUnitAsyncMode

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

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

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

4.98 Parameter SpiJobStartNotificationEnable

settings.

Note This feature is a SpiAutosarExt feature to enable the start job notification.

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

4.99 Parameter SpiDisableDemReportErrorStatus

SpiDisableDemReportErrorStatus

Switches the Diagnostic Error Reporting and Notification OFF.

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

4.100 Parameter SpiFlexioEnable

SpiFlexioEnable

If it is true, FLEXIO feature is enabled

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



Chapter 5

Module Index

5.1 Software Specification

Here is a list of all modules:

Flexio_Spi Driver	96
Lpspi Driver	108
Spi Driver	126

Chapter 6

Module Documentation

6.1 Flexio_Spi Driver

6.1.1 Detailed Description

Data Structures

- struct [Flexio_Spi_Ip_DeviceParamsType](#)
Structure defining some parameters often change of the spi bus. [More...](#)
- struct [Flexio_Spi_Ip_ExternalDeviceType](#)
Structure defining the parameters of the spi bus. [More...](#)
- struct [Flexio_Spi_Ip_ConfigType](#)
Structure defining information needed for SPI driver initialization. [More...](#)
- struct [Flexio_Spi_Ip_StateStructureType](#)
Structure defining information needed for internal state of the driver. [More...](#)

Macros

- `#define SPI_STOP_SEC_CONFIG_DATA_UNSPECIFIED`
Export Post-Build configurations.

Types Reference

- typedef void(* [Flexio_Spi_Ip_CallbackType](#)) (uint8 Instance, [Flexio_Spi_Ip_EventType](#) Event)
Callback for all peripherals which supports SPI features.

Enum Reference

- enum [Flexio_Spi_Ip_EventType](#)
Enum defining the possible events which triggers end of transfer callback.
- enum [Flexio_Spi_Ip_ModeType](#)
Enum defining the possible transfer modes.
- enum [Flexio_Spi_Ip_HwStatusType](#)
Enum defining the possible states of SPI/DSPI hardware unit.
- enum [Flexio_Spi_Ip_StatusType](#)
Enum defining the possible return types.

Function Reference

- [Flexio_Spi_Ip_StatusType Flexio_Spi_Ip_Init](#) (const [Flexio_Spi_Ip_ConfigType](#) *PhyUnitConfigPtr)
FLEXIO_SPI peripheral initialization.
- [Flexio_Spi_Ip_StatusType Flexio_Spi_Ip_DeInit](#) (uint8 Instance)
FLEXIO_SPI peripheral deinitialization.
- [Flexio_Spi_Ip_StatusType Flexio_Spi_Ip_SyncTransmit](#) (const [Flexio_Spi_Ip_ExternalDeviceType](#) *ExternalDevice, const uint8 *TxBuffer, uint8 *RxBuffer, uint16 Length, uint32 TimeOut)
FLEXIO_SPI synchronous transmission.
- [Flexio_Spi_Ip_StatusType Flexio_Spi_Ip_AsyncTransmit](#) (const [Flexio_Spi_Ip_ExternalDeviceType](#) *ExternalDevice, const uint8 *TxBuffer, uint8 *RxBuffer, uint16 Length, [Flexio_Spi_Ip_CallbackType](#) EndCallback)
FLEXIO_SPI asynchronous transmission.
- [Flexio_Spi_Ip_HwStatusType Flexio_Spi_Ip_GetStatus](#) (uint8 Instance)
Get status of HW unit.
- void [Flexio_Spi_Ip_ManageBuffers](#) (uint8 Instance)
Process transfer in POLLING mode.
- [Flexio_Spi_Ip_StatusType Flexio_Spi_Ip_UpdateFrameSize](#) (const [Flexio_Spi_Ip_ExternalDeviceType](#) *ExternalDevice, uint8 FrameSize)
FLEXIO_SPI change frame size.
- [Flexio_Spi_Ip_StatusType Flexio_Spi_Ip_UpdateLsb](#) (const [Flexio_Spi_Ip_ExternalDeviceType](#) *ExternalDevice, boolean Lsb)
FLEXIO_SPI change bit order.
- [Flexio_Spi_Ip_StatusType Flexio_Spi_Ip_UpdateDefaultTransmitData](#) (const [Flexio_Spi_Ip_ExternalDeviceType](#) *ExternalDevice, uint32 DefaultData)
FLEXIO_SPI change default transmit data.
- [Flexio_Spi_Ip_StatusType Flexio_Spi_Ip_UpdateTransferMode](#) (uint8 Instance, [Flexio_Spi_Ip_ModeType](#) Mode)
FLEXIO_SPI change transfer mode.
- void [Flexio_Spi_Ip_Cancel](#) (uint8 Instance)
FLEXIO_SPI cancel current transmission.

6.1.2 Data Structure Documentation

6.1.2.1 struct [Flexio_Spi_Ip_DeviceParamsType](#)

Structure defining some parameters often change of the spi bus.

Definition at line 151 of file [Flexio_Spi_Ip_Types.h](#).

Data Fields

Type	Name	Description
uint8	FrameSize	Frame size configured
boolean	Lsb	Transfer LSB first or MSB first
uint32	DefaultData	Default data to send when TxBuffer is NULL_PTR

6.1.2.2 struct Flexio_Spi_Ip_ExternalDeviceType

Structure defining the parameters of the spi bus.

Definition at line 159 of file Flexio_Spi_Ip_Types.h.

Data Fields

	Type	Name	Description
	uint8	Instance	Instance of the hardware unit.
	uint8	Cpol	
	uint8	Cpha	
	uint32	ClkTimeCmpBaudRate	
	uint32	ClkTimeCfgTimDec	
	uint32	TxShiftCtl	SHIFTCTL register of TX
	uint32	TxShiftCfg	SHIFTCFG register of TX
	uint32	RxShiftCtl	SHIFTCTL register of RX
	uint32	RxShiftCfg	SHIFTCFG register of RX
	uint32	ClkTimeCmp	TIMCMP register of CLK
	uint32	ClkTimeCfg	TIMCFG register of CLK
	uint32	ClkTimeCtl	TIMCTL register of CLK
	uint32	CsTimeCmp	TIMCMP register of CS
	uint32	CsTimeCfg	TIMCFG register of CS
	uint32	CsTimeCtl	TIMCTL register of CS
Flexio_Spi_Ip_DeviceParamsType *		DeviceParams	Contain configuration for bit order, frame size, default transmit data.

6.1.2.3 struct Flexio_Spi_Ip_ConfigType

Structure defining information needed for SPI driver initialization.

Definition at line 185 of file Flexio_Spi_Ip_Types.h.

Data Fields

Type	Name	Description
uint8	Instance	Instance of the hardware unit.

Data Fields

Type	Name	Description
boolean	DmaUsed	DMA is used or not
uint8	TxDmaChannel	Id of TX DMA channel for transmission
uint8	RxDmaChannel	Id of RX DMA channel for receive
Flexio_Spi_Ip_ModeType	TransferMode	Transfer mode for HWunit
uint32	FrameSize	Frame size configured
boolean	Lsb	Transfer LSB first or MSB first
uint32	DefaultData	Default data to send when TxBuffer is NULL_PTR
uint8	TxShifterIndex	No. of shifter for TX
uint8	RxShifterIndex	No. of shifter for RX
uint8	ClkTimerIndex	No. of timer for CLK
uint8	CsTimerIndex	No. of timer for CS
uint8	StateIndex	State of current transfer

6.1.2.4 struct Flexio_Spi_Ip_StateStructureType

Structure defining information needed for internal state of the driver.

Definition at line 208 of file Flexio_Spi_Ip_Types.h.

Data Fields

Type	Name	Description
Flexio_Spi_Ip_ModeType	TransferMode	Store current transfer mode for HWunit
boolean	FirstChannel	This is the first channel in a job
Flexio_Spi_Ip_HwStatusType	Status	0 = available, 1 = busy, 2 = fail due to overflow or underflow
uint8 *	RxBuffer	Store pointer for Rx buffer
const uint8 *	TxBuffer	Store pointer for Tx buffer
Flexio_Spi_Ip_CallbackType	Callback	Store pointer for call back function
uint16	RxIndex	Store current Rx index to receive data in Rx buffer
uint16	TxIndex	Store current Tx index to transmit data in Tx buffer
uint16	ExpectedFifoReads	Store number of frames needs to be receive for current transfer
uint16	ExpectedFifoWrites	Store number of frames needs to be transmit for current transfer
const Flexio_Spi_Ip_ConfigType *	PhyUnitConfig	
const Flexio_Spi_Ip_ExternalDeviceType *	ExternalDevice	Store externalDevice

6.1.3 Macro Definition Documentation

6.1.3.1 SPI_STOP_SEC_CONFIG_DATA_UNSPECIFIED

```
#define SPI_STOP_SEC_CONFIG_DATA_UNSPECIFIED
```

Export Post-Build configurations.

Definition at line 116 of file Flexio_Spi_Ip.h.

6.1.4 Types Reference

6.1.4.1 Flexio_Spi_Ip_CallbackType

```
typedef void(* Flexio_Spi_Ip_CallbackType) (uint8 Instance, Flexio_Spi_Ip_EventType Event)
```

Callback for all peripherals which supports SPI features.

Definition at line 106 of file Flexio_Spi_Ip_Types.h.

6.1.5 Enum Reference

6.1.5.1 Flexio_Spi_Ip_EventType

```
enum Flexio_Spi_Ip_EventType
```

Enum defining the possible events which triggers end of transfer callback.

Enumerator

FLEXIO_SPI_IP_EVENT_END_TRANSFER	The transfer is successfully done.
FLEXIO_SPI_IP_EVENT_FAULT	The transfer failed due to overflow/underflow.

Definition at line 99 of file Flexio_Spi_Ip_Types.h.

6.1.5.2 Flexio_Spi_Ip_ModeType

```
enum Flexio_Spi_Ip_ModeType
```

Enum defining the possible transfer modes.

Enumerator

FLEXIO_SPI_IP_POLLING	For polling mode the application must call periodically Spi_Ip_ManageBuffers after asynchronous transfers.
FLEXIO_SPI_IP_INTERRUPT	For interrupt mode the application doesn't need to perform any additional operations after asynchronous transfers.

Definition at line 111 of file Flexio_Spi_Ip_Types.h.

6.1.5.3 Flexio_Spi_Ip_HwStatusType

```
enum Flexio_Spi_Ip_HwStatusType
```

Enum defining the possible states of SPI/DSPI hardware unit.

Enumerator

FLEXIO_SPI_IP_UNINIT	Module is not initialized.
FLEXIO_SPI_IP_IDLE	Module is not used.
FLEXIO_SPI_IP_BUSY	A transfer is in progress.
FLEXIO_SPI_IP_FAULT	During last transfer a fault occurred.

Definition at line 120 of file Flexio_Spi_Ip_Types.h.

6.1.5.4 Flexio_Spi_Ip_StatusType

```
enum Flexio_Spi_Ip_StatusType
```

Enum defining the possible return types.

Enumerator

FLEXIO_SPI_IP_STATUS_SUCCESS	Successful operation.
FLEXIO_SPI_IP_STATUS_FAIL	Failed operation.
FLEXIO_SPI_IP_FIFO_ERROR	Overflow or underflow error.
FLEXIO_SPI_IP_TIMEOUT	Timeout error.

Definition at line 131 of file Flexio_Spi_Ip_Types.h.

6.1.6 Function Reference

6.1.6.1 Flexio_Spi_Ip_Init()

```
Flexio_Spi_Ip_StatusType Flexio_Spi_Ip_Init (
    const Flexio_Spi_Ip_ConfigType * PhyUnitConfigPtr )
```

FLEXIO_SPI peripheral initialization.

The function initialize the SPI Unit specified in the configuration.

Parameters

in	<i>PhyUnitConfigPtr</i>	- pointer to the specified SPI Unit configuration.
----	-------------------------	--

Returns

FLEXIO_SPI_IP_STATUS_SUCCESS: Initialization command has been accepted. FLEXIO_SPI_IP_STATUS_FAIL: Initialization command has not been accepted.

6.1.6.2 Flexio_Spi_Ip_DeInit()

```
Flexio_Spi_Ip_StatusType Flexio_Spi_Ip_DeInit (
    uint8 Instance )
```

FLEXIO_SPI peripheral deinitialization.

The function de-initialize the SPI peripheral instance specified. All registers of SPI peripheral will be reset.

Parameters

in	<i>Instance</i>	- SPI peripheral instance number.
----	-----------------	-----------------------------------

Returns

FLEXIO_SPI_IP_STATUS_SUCCESS: De-initialization command has been accepted. FLEXIO_SPI_IP_STATUS_FAIL: De-initialization command has not been accepted.

6.1.6.3 Flexio_Spi_Ip_SyncTransmit()

```
Flexio_Spi_Ip_StatusType Flexio_Spi_Ip_SyncTransmit (
    const Flexio_Spi_Ip_ExternalDeviceType * ExternalDevice,
    const uint8 * TxBuffer,
```

```
uint8 * RxBuffer,
uint16 Length,
uint32 TimeOut )
```

FLEXIO_SPI synchronous transmission.

This function initializes a synchronous transfer using the bus parameters provided by external device.

Parameters

in	<i>ExternalDevice</i>	- pointer to the external device where data is transmitted.
in	<i>TxBuffer</i>	- pointer to transmit buffer.
	<i>[in-out]</i>	RxBuffer - pointer to receive buffer.
in	<i>Length</i>	- number of bytes to be sent.
in	<i>TimeOut</i>	- duration for sending one frame.

Returns

FLEXIO_SPI_IP_STATUS_SUCCESS: Transmission command has been accepted. FLEXIO_SPI_IP_↔
FIFO_ERROR: Overflow or underflow error occurred. FLEXIO_SPI_IP_STATUS_FAIL: Transmission
command has not been accepted. FLEXIO_SPI_IP_TIMEOUT: Timeout error occurred.

6.1.6.4 Flexio_Spi_Ip_AsyncTransmit()

```
Flexio_Spi_Ip_StatusType Flexio_Spi_Ip_AsyncTransmit (
    const Flexio_Spi_Ip_ExternalDeviceType * ExternalDevice,
    const uint8 * TxBuffer,
    uint8 * RxBuffer,
    uint16 Length,
    Flexio_Spi_Ip_CallbackType EndCallback )
```

FLEXIO_SPI asynchronous transmission.

This function initializes an asynchronous transfer using the bus parameters provided by external device. After Flexio_Spi_Ip_Init function is called, FLEXIO_SPI_IP_POLLING mode is set as default to change the default mode Flexio_Spi_Ip_UpdateTransferMode should be called.

Parameters

in	<i>ExternalDevice</i>	- pointer to the external device where data is transmitted
in	<i>TxBuffer</i>	- pointer to transmit buffer.
	<i>[in-out]</i>	RxBuffer - pointer to receive buffer.
in	<i>Length</i>	- number of bytes to be sent.
in	<i>Callback</i>	- callback function is called at the end of transfer.

Returns

FLEXIO_SPI_IP_STATUS_SUCCESS: Transmission command has been accepted. FLEXIO_SPI_IP_STATUS_FAIL: Transmission command has not been accepted.

6.1.6.5 Flexio_Spi_Ip_GetStatus()

```
Flexio_Spi_Ip_HwStatusType Flexio_Spi_Ip_GetStatus (
    uint8 Instance )
```

Get status of HW unit.

This function will return status of HW unit assigned.

Parameters

in	<i>instance</i>	Instance of the hardware unit.
----	-----------------	--------------------------------

Returns

Flexio_Spi_Ip_HwStatusType

Return values

<i>FLEXIO_SPI_IP_IDLE</i>	Hardware unit is not used
<i>FLEXIO_SPI_IP_BUSY</i>	A transfer is in progress
<i>FLEXIO_SPI_IP_FAULT</i>	During last transfer a fault occurred

6.1.6.6 Flexio_Spi_Ip_ManageBuffers()

```
void Flexio_Spi_Ip_ManageBuffers (
    uint8 Instance )
```

Process transfer in POLLING mode.

This function shall polls the SPI interrupts linked to SPI peripheral instance allocated to the transmission of data to enable the evolution of transmission state machine.

Parameters

in	<i>Instance</i>	- SPI peripheral instance number.
----	-----------------	-----------------------------------

Returns

void

6.1.6.7 Flexio_Spi_Ip_UpdateFrameSize()

```
Flexio_Spi_Ip_StatusType Flexio_Spi_Ip_UpdateFrameSize (
    const Flexio_Spi_Ip_ExternalDeviceType * ExternalDevice,
    uint8 FrameSize )
```

FLEXIO_SPI change frame size.

This function updates frame size of specific external device configuration for next transfers.

Parameters

in	<i>ExternalDevice</i>	- pointer to the external device configuration.
in	<i>FrameSize</i>	- Frame size.

Returns

FLEXIO_SPI_IP_STATUS_SUCCESS: Setting command has been accepted. FLEXIO_SPI_IP_STATUS_FAIL: Setting command has not been accepted.

6.1.6.8 Flexio_Spi_Ip_UpdateLsb()

```
Flexio_Spi_Ip_StatusType Flexio_Spi_Ip_UpdateLsb (
    const Flexio_Spi_Ip_ExternalDeviceType * ExternalDevice,
    boolean Lsb )
```

FLEXIO_SPI change bit order.

This function updates bits order LSB or MSB of specific external device configuration for next transfer.

Parameters

in	<i>ExternalDevice</i>	- pointer to the external device configuration.
in	<i>Lsb</i>	- Data is transferred LSB first or not.

Returns

FLEXIO_SPI_IP_STATUS_SUCCESS: Setting command has been accepted. FLEXIO_SPI_IP_STATUS_FAIL: Setting command has not been accepted.

6.1.6.9 Flexio_Spi_Ip_UpdateDefaultTransmitData()

```
Flexio_Spi_Ip_StatusType Flexio_Spi_Ip_UpdateDefaultTransmitData (
    const Flexio_Spi_Ip_ExternalDeviceType * ExternalDevice,
    uint32 DefaultData )
```

FLEXIO_SPI change default transmit data.

This function updates default transmit data of specific external device configuration for next transfer.

Parameters

in	<i>ExternalDevice</i>	- pointer to the external device configuration.
in	<i>DefaultData</i>	- New default transmit data.

Returns

FLEXIO_SPI_IP_STATUS_SUCCESS: Setting command has been accepted. FLEXIO_SPI_IP_STAT←
US_FAIL: Setting command has not been accepted.

6.1.6.10 Flexio_Spi_Ip_UpdateTransferMode()

```
Flexio_Spi_Ip_StatusType Flexio_Spi_Ip_UpdateTransferMode (
    uint8 Instance,
    Flexio_Spi_Ip_ModeType Mode )
```

FLEXIO_SPI change transfer mode.

This function updates the asynchronous mechanism mode for the specified SPI Hardware microcontroller peripheral.

Parameters

in	<i>Instance</i>	- SPI peripheral instance number.
in	<i>Mode</i>	- new mode (interrupt or polling).

Returns

FLEXIO_SPI_IP_STATUS_SUCCESS: Setting command has been accepted. FLEXIO_SPI_IP_STAT←
US_FAIL: Setting command has not been accepted.

6.1.6.11 Flexio_Spi_Ip_Cancel()

```
void Flexio_Spi_Ip_Cancel (
    uint8 Instance )
```

FLEXIO_SPI cancel current transmission.

This function will cancel current asynchronous transmission.

Parameters

in	<i>Instance</i>	Instance of the hardware unit.
----	-----------------	--------------------------------

6.2 Lpspi Driver

6.2.1 Detailed Description

Data Structures

- struct [Lpspi_Ip_DeviceParamsType](#)
Structure defining some parameters often change of the spi bus. [More...](#)
- struct [Lpspi_Ip_ExternalDeviceType](#)
Structure defining the parameters of the spi bus. [More...](#)
- struct [Lpspi_Ip_CmdDmaFastType](#)
Structure defining transimtion command needed for Dma Fast transfer. [More...](#)
- struct [Lpspi_Ip_FastTransferType](#)
Structure defining information needed for Dma Fast transfer session. [More...](#)
- struct [Lpspi_Ip_ConfigType](#)
Structure defining information needed for SPI driver initialization. [More...](#)
- struct [Lpspi_Ip_StateStructureType](#)
Structure defining information needed for internal state of the driver. [More...](#)

Types Reference

- typedef void(* [Lpspi_Ip_CallbackType](#)) (uint8 Instance, [Lpspi_Ip_EventType](#) Event)
Callback for all peripherals which supports SPI features.

Enum Reference

- enum [Lpspi_Ip_EventType](#)
Enum defining the possible events which triggers end of transfer callback.
- enum [Lpspi_Ip_ModeType](#)
Enum defining the possible transfer modes.
- enum [Lpspi_Ip_HwStatusType](#)
Enum defining the possible states of SPI/DSPI hardware unit.
- enum [Lpspi_Ip_StatusType](#)
Enum defining the possible return types.
- enum [Lpspi_Ip_HalfDuplexType](#)
Enum defining the half duplex types.

Function Reference

- [Lpspi_Ip_StatusType Lpspi_Ip_Init](#) (const [Lpspi_Ip_ConfigType](#) *PhyUnitConfigPtr)
LPSPI peripheral initialization.
- [Lpspi_Ip_StatusType Lpspi_Ip_DeInit](#) (uint8 Instance)
LPSPI peripheral deinitialization.
- [Lpspi_Ip_StatusType Lpspi_Ip_SyncTransmit](#) (const [Lpspi_Ip_ExternalDeviceType](#) *ExternalDevice, const uint8 *TxBuffer, uint8 *RxBuffer, uint16 Length, uint32 TimeOut)
LPSPI synchronous transmission.
- [Lpspi_Ip_StatusType Lpspi_Ip_AsyncTransmit](#) (const [Lpspi_Ip_ExternalDeviceType](#) *ExternalDevice, const uint8 *TxBuffer, uint8 *RxBuffer, uint16 Length, [Lpspi_Ip_CallbackType](#) EndCallback)
LPSPI asynchronous transmission.
- [Lpspi_Ip_StatusType Lpspi_Ip_SyncTransmitHalfDuplex](#) (const [Lpspi_Ip_ExternalDeviceType](#) *ExternalDevice, uint8 *Buffer, uint16 Length, [Lpspi_Ip_HalfDuplexType](#) TransferType, uint32 TimeOut)
LPSPI synchronous transmission support half duplex mode.
- [Lpspi_Ip_StatusType Lpspi_Ip_AsyncTransmitHalfDuplex](#) (const [Lpspi_Ip_ExternalDeviceType](#) *ExternalDevice, uint8 *Buffer, uint16 Length, [Lpspi_Ip_HalfDuplexType](#) TransferType, [Lpspi_Ip_CallbackType](#) EndCallback)
LPSPI asynchronous transmission support half duplex mode.
- [Lpspi_Ip_StatusType Lpspi_Ip_AsyncTransmitFast](#) (const [Lpspi_Ip_FastTransferType](#) *FastTransferCfg, uint8 NumberOfTransfer, [Lpspi_Ip_CallbackType](#) EndCallback)
LPSPI asynchronous transmission fast.
- [Lpspi_Ip_HwStatusType Lpspi_Ip_GetStatus](#) (uint8 Instance)
Get status of HW unit.
- void [Lpspi_Ip_ManageBuffers](#) (uint8 Instance)
Process transfer in POLLING mode.
- [Lpspi_Ip_StatusType Lpspi_Ip_UpdateFrameSize](#) (const [Lpspi_Ip_ExternalDeviceType](#) *ExternalDevice, uint16 FrameSize)
LPSPI change frame size.
- [Lpspi_Ip_StatusType Lpspi_Ip_UpdateLsb](#) (const [Lpspi_Ip_ExternalDeviceType](#) *ExternalDevice, boolean Lsb)
LPSPI change bit order.
- [Lpspi_Ip_StatusType Lpspi_Ip_UpdateDefaultTransmitData](#) (const [Lpspi_Ip_ExternalDeviceType](#) *ExternalDevice, uint32 DefaultData)
LPSPI change default transmit data.
- [Lpspi_Ip_StatusType Lpspi_Ip_UpdateTransferMode](#) (uint8 Instance, [Lpspi_Ip_ModeType](#) Mode)
LPSPI change transfer mode.
- void [Lpspi_Ip_Cancel](#) (uint8 Instance)
LPSPI cancel current asynchronous transmission.

Variables

- const [Lpspi_Ip_ExternalDeviceType Lpspi_Ip_DeviceAttributes_SpiExternalDevice_LPSPi0_VS_0](#)
Export Post-Build configurations.

6.2.2 Data Structure Documentation

6.2.2.1 struct Lpspi_Ip_DeviceParamsType

Structure defining some parameters often change of the spi bus.

Definition at line 161 of file Lpspi_Ip_Types.h.

Data Fields

Type	Name	Description
uint16	FrameSize	Frame size configured
boolean	Lsb	Transfer LSB first or MSB first
uint32	DefaultData	Default data to send when TxBuffer is NULL_PTR
Lpspi_Ip_HalfDuplexType	TransferType	TransferType

6.2.2.2 struct Lpspi_Ip_ExternalDeviceType

Structure defining the parameters of the spi bus.

Definition at line 172 of file Lpspi_Ip_Types.h.

Data Fields

Type	Name	Description
uint8	Instance	Instance of the hardware unit.
uint32	Ccr	CCR register which contains clocking and frame size configuration.
uint32	Tcr	TCR register which contains clock polarities, frame size, which PCS and continuous mode.
uint32	HalfDuplexCfgr1	CFGR1 register which contains bit fields to support half duplex mode .
Lpspi_Ip_DeviceParamsType *	DeviceParams	Contain configuration for bit order, frame size, default transmit data.

6.2.2.3 struct Lpspi_Ip_CmdDmaFastType

Structure defining transmtion command needed for Dma Fast transfer.

Definition at line 202 of file Lpspi_Ip_Types.h.

Data Fields

Type	Name	Description
uint32	DmaFastTcrCmd	Contains transfer command for Dma Fast transfer.
uint32	DmaFastTcrCmdLast	Contains transfer command and disable continuos mode for Dma Fast transfer.
uint32	DefaultData	Default data to send when TxBuffer is NULL_PTR

6.2.2.4 struct Lpspi_Ip_FastTransferType

Structure defining information needed for Dma Fast transfer session.

Definition at line 212 of file Lpspi_Ip_Types.h.

Data Fields

- const [Lpspi_Ip_ExternalDeviceType](#) * [ExternalDevice](#)
- const uint8 * [TxBuffer](#)
- uint8 * [RxBuffer](#)
- uint32 [DefaultData](#)
- uint16 [Length](#)
- boolean [KeepCs](#)

6.2.2.4.1 Field Documentation

6.2.2.4.1.1 ExternalDevice `const Lpspi_Ip_ExternalDeviceType* ExternalDevice`

Point to external device configuration

Definition at line 214 of file Lpspi_Ip_Types.h.

6.2.2.4.1.2 TxBuffer `const uint8* TxBuffer`

Store pointer for Tx buffer

Definition at line 215 of file Lpspi_Ip_Types.h.

6.2.2.4.1.3 RxBuffer `uint8* RxBuffer`

Store pointer for Rx buffer

Definition at line 216 of file Lpspi_Ip_Types.h.

6.2.2.4.1.4 DefaultData `uint32 DefaultData`

Default data to send when TxBuffer is NULL_PTR

Definition at line 217 of file Lpspi_Ip_Types.h.

6.2.2.4.1.5 Length `uint16 Length`

Number of bytes to be sent

Definition at line 218 of file `Lpspi_Ip_Types.h`.

6.2.2.4.1.6 KeepCs `boolean KeepCs`

Keep CS signal after transfer session completed

Definition at line 219 of file `Lpspi_Ip_Types.h`.

6.2.2.5 struct `Lpspi_Ip_ConfigType`

Structure defining information needed for SPI driver initialization.

Definition at line 225 of file `Lpspi_Ip_Types.h`.

Data Fields

Type	Name	Description
uint8	Instance	Instance of the hardware unit.
uint32	Cr	It contains only debug enable.
uint32	Cfgr1	It contains PCS polarities.
boolean	SlaveMode	
boolean	DmaUsed	DMA is used or not
uint8	TxDmaChannel	Id of TX DMA channel for transmtion
uint8	RxDmaChannel	Id of RX DMA channel for receive
uint8	MaxNumOfFastTransfer	Maximum number of transfers in Dma Fast
Lpspi_Ip_CmdDmaFastType *	CmdDmaFast	Point to list of TCR command used in Dma Fast transfer
uint8	NumberTxSG	Number of TCD Scatter Gather for Tx DMA channel used in Dma Fast transfer
uint8	NumberRxSG	Number of TCD Scatter Gather for Rx DMA channel used in Dma Fast transfer
const uint8 *	TxDmaFastSGId	Point to list of TCD Scatter Gather Id for Tx DMA channel used in Dma Fast transfer
const uint8 *	RxDmaFastSGId	Point to list of TCD Scatter Gather Id for Rx DMA channel used in Dma Fast transfer
Lpspi_Ip_ModeType	TransferMode	Transfer mode for HWunit
uint8	StateIndex	State of current transfer

6.2.2.6 struct Lpspi_Ip_StateStructureType

Structure defining information needed for internal state of the driver.

Definition at line 251 of file Lpspi_Ip_Types.h.

Data Fields

Type	Name	Description
Lpspi_Ip_ModeType	TransferMode	Store current transfer mode for HWunit
Lpspi_Ip_HwStatusType	Status	0 = available, 1 = busy, 2 = fail due to overflow or underflow
uint8 *	RxBuffer	Store pointer for Rx buffer
const uint8 *	TxBuffer	Store pointer for Tx buffer
Lpspi_Ip_CallbackType	Callback	Store pointer for call back function
uint16	RxIndex	Store current Rx index to receive data in Rx buffer
uint16	TxIndex	Store current Tx index to transmit data in Tx buffer
uint16	ExpectedFifoReads	Store number of words needed to be receive for current transfer
uint16	ExpectedFifoWrites	Store number of words needed to be transmitted for current transfer
boolean	KeepCs	Keep CS signal after transfers completed.
boolean	FirstCmd	
const Lpspi_Ip_ConfigType *	PhyUnitConfig	
const Lpspi_Ip_ExternalDeviceType *	ExternalDevice	
uint16	TxFrameSize	Store current frame size for HWunit
boolean	TxLsb	Transfer LSB first or MSB first
const uint8 *	TxBufferNext	Store pointer for Tx buffer
uint16	FrameSizeNext	Store current frame size for HWunit
boolean	LsbNext	Transfer LSB first or MSB first
uint32	DefaultDataNext	Default data to send when TxBuffer is NULL_PTR
uint16	LengthNext	Store number of frames needs to be transmit for current transfer
boolean	NextTransferConfigAvailable	Flag to check next transfer configuration is available
boolean	NextTransferDone	Flag to check next transfer done
uint8	CurrentTxFifoSlot	Number of TX FIFO slots are current available.
uint32	HalfDuplexTcrCommand	Save the value which will be written to TCR register in DMA mode
boolean	NextChannelsRX	Save the value which will be written to TCR register in DMA mode
boolean	TxDoneFlag	Flag to check TX is done

6.2.3 Types Reference

6.2.3.1 Lpspi_Ip_CallbackType

```
typedef void(* Lpspi_Ip_CallbackType) (uint8 Instance, Lpspi_Ip_EventType Event)
```

Callback for all peripherals which supports SPI features.

Definition at line 115 of file Lpspi_Ip_Types.h.

6.2.4 Enum Reference

6.2.4.1 Lpspi_Ip_EventType

```
enum Lpspi_Ip_EventType
```

Enum defining the possible events which triggers end of transfer callback.

Enumerator

LPSPi_IP_EVENT_END_TRANSFER	The transfer is successfully done.
LPSPi_IP_EVENT_FAULT	The transfer failed due to overflow/underflow.

Definition at line 108 of file Lpspi_Ip_Types.h.

6.2.4.2 Lpspi_Ip_ModeType

```
enum Lpspi_Ip_ModeType
```

Enum defining the possible transfer modes.

Enumerator

LPSPi_IP_POLLING	For polling mode the application must call periodically Spi_Ip_ManageBuffers after asynchronous transfers.
LPSPi_IP_INTERRUPT	For interrupt mode the application doesn't need to perform any additional operations after asynchronous transfers. The application must enable the interrupt requests and install the right callbacks.

Definition at line 120 of file Lpspi_Ip_Types.h.

6.2.4.3 Lpspi_Ip_HwStatusType

enum `Lpspi_Ip_HwStatusType`

Enum defining the possible states of SPI/DSPI hardware unit.

Enumerator

LPSP_I_P_UNINIT	Module is not initialized.
LPSP_I_P_IDLE	Module is not used.
LPSP_I_P_BUSY	A transfer is in progress.
LPSP_I_P_FAULT	During last transfer a fault occurred.

Definition at line 130 of file `Lpspi_Ip_Types.h`.

6.2.4.4 Lpspi_Ip_StatusType

enum `Lpspi_Ip_StatusType`

Enum defining the possible return types.

Enumerator

LPSP_I_P_STATUS_SUCCESS	Successful operation.
LPSP_I_P_STATUS_FAIL	Failed operation.
LPSP_I_P_FIFO_ERROR	Overflow or underflow error.
LPSP_I_P_TIMEOUT	Timeout error.

Definition at line 141 of file `Lpspi_Ip_Types.h`.

6.2.4.5 Lpspi_Ip_HalfDuplexType

enum `Lpspi_Ip_HalfDuplexType`

Enum defining the half duplex types.

Enumerator

LPSP_I_P_HALF_DUPLEX_TRANSMIT	Transmit only.
LPSP_I_P_HALF_DUPLEX_RECEIVE	Receive only.
LPSP_I_P_FULL_DUPLEX	Full duplex mode.

Definition at line 152 of file Lpspi_Ip_Types.h.

6.2.5 Function Reference

6.2.5.1 Lpspi_Ip_Init()

```
Lpspi_Ip_StatusType Lpspi_Ip_Init (
    const Lpspi_Ip_ConfigType * PhyUnitConfigPtr )
```

LPSPI peripheral initialization.

The function initialize the SPI Unit specified in the configuration.

Parameters

in	<i>PhyUnitConfigPtr</i>	- pointer to the specified SPI Unit configuration.
----	-------------------------	--

Returns

LPSPI_IP_STATUS_SUCCESS: Initialization command has been accepted. LPSPI_IP_STATUS_FAIL: Initialization command has not been accepted.

6.2.5.2 Lpspi_Ip_DeInit()

```
Lpspi_Ip_StatusType Lpspi_Ip_DeInit (
    uint8 Instance )
```

LPSPI peripheral deinitialization.

The function de-initialize the SPI peripheral instance specified. All registers of SPI peripheral will be reset.

Parameters

in	<i>Instance</i>	- SPI peripheral instance number.
----	-----------------	-----------------------------------

Returns

LPSPI_IP_STATUS_SUCCESS: De-initialization command has been accepted. LPSPI_IP_STATUS_FAIL: De-initialization command has not been accepted.

6.2.5.3 Lpspi_Ip_SyncTransmit()

```
Lpspi_Ip_StatusType Lpspi_Ip_SyncTransmit (
    const Lpspi_Ip_ExternalDeviceType * ExternalDevice,
    const uint8 * TxBuffer,
    uint8 * RxBuffer,
    uint16 Length,
    uint32 TimeOut )
```

LPSPI synchronous transmission.

This function initializes a synchronous transfer using the bus parameters provided by external device. Depending on the FrameSize TxBuffer and RxBuffer are handled in the following way:

- 8: Array is interpreted as uint8
- 16: Array is interpreted as uint16
- 32: Array is interpreted as uint32
- >32: Array is interpreted as uint32 and if Length is not multiple of 4 the last bytes of the frame, until a multiple of 4, are ignored.

Parameters

in	<i>ExternalDevice</i>	- pointer to the external device where data is transmitted.
in	<i>TxBuffer</i>	- pointer to transmit buffer.
	<i>[in-out]</i>	RxBuffer - pointer to receive buffer.
in	<i>Length</i>	- number of bytes to be sent.
in	<i>TimeOut</i>	- duration for sending one frame.

Returns

LPSPI_IP_STATUS_SUCCESS: Transmission command has been accepted. LPSPI_IP_FIFO_ERROR←: Overflow or underflow error occurred. LPSPI_IP_STATUS_FAIL: Transmission command has not been accepted. LPSPI_IP_TIMEOUT: Timeout error occurred.

6.2.5.4 Lpspi_Ip_AsyncTransmit()

```
Lpspi_Ip_StatusType Lpspi_Ip_AsyncTransmit (
    const Lpspi_Ip_ExternalDeviceType * ExternalDevice,
    const uint8 * TxBuffer,
    uint8 * RxBuffer,
    uint16 Length,
    Lpspi_Ip_CallbackType EndCallback )
```

LPSPI asynchronous transmission.

Module Documentation

This function initializes an asynchronous transfer using the bus parameters provided by external device. After Lpspi_Ip_Init function is called, LPSPI_IP_POLLING mode is set as default to change the default mode Lpspi_Ip_UpdateTransferMode should be called.

Parameters

in	<i>ExternalDevice</i>	- pointer to the external device where data is transmitted
in	<i>TxBuffer</i>	- pointer to transmit buffer.
	<i>[in-out]</i>	RxBuffer - pointer to receive buffer.
in	<i>Length</i>	- number of bytes to be sent.
in	<i>EndCallback</i>	- callback function is called at the end of transfer.

Returns

LPSPI_IP_STATUS_SUCCESS: Transmission command has been accepted. LPSPI_IP_STATUS_FAIL: Transmission command has not been accepted.

6.2.5.5 Lpspi_Ip_SyncTransmitHalfDuplex()

```
Lpspi_Ip_StatusType Lpspi_Ip_SyncTransmitHalfDuplex (
    const Lpspi_Ip_ExternalDeviceType * ExternalDevice,
    uint8 * Buffer,
    uint16 Length,
    Lpspi_Ip_HalfDuplexType TransferType,
    uint32 TimeOut )
```

LPSPI synchronous transmission support half duplex mode.

This function initializes a synchronous transfer using the bus parameters provided by external device.

Parameters

in	<i>ExternalDevice</i>	- pointer to the external device where data is transmitted.
in	<i>Buffer</i>	- pointer to transmit buffer.
in	<i>Length</i>	- number of bytes to be sent.
in	<i>TimeOut</i>	- duration for sending one frame.

Returns

LPSPI_IP_STATUS_SUCCESS: Transmission command has been accepted. LPSPI_IP_FIFO_ERROR: Overflow or underflow error occurred. LPSPI_IP_STATUS_FAIL: Transmission command has not been accepted. LPSPI_IP_TIMEOUT: Timeout error occurred.

6.2.5.6 Lpspi_Ip_AsyncTransmitHalfDuplex()

```
Lpspi_Ip_StatusType Lpspi_Ip_AsyncTransmitHalfDuplex (
    const Lpspi_Ip_ExternalDeviceType * ExternalDevice,
```

```
uint8 * Buffer,
uint16 Length,
Lpspi_Ip_HalfDuplexType TransferType,
Lpspi_Ip_CallbackType EndCallback )
```

LPSPI asynchronous transmission support half duplex mode.

This function initializes an asynchronous transfer using the bus parameters provided by external device.

Parameters

in	<i>ExternalDevice</i>	- pointer to the external device where data is transmitted
in	<i>Buffer</i>	- pointer to transmit buffer.
in	<i>Length</i>	- number of bytes to be sent.
in	<i>EndCallback</i>	- callback function is called at the end of transfer.

Returns

LPSPI_IP_STATUS_SUCCESS: Transmission command has been accepted. LPSPI_IP_STATUS_FAIL: Transmission command has not been accepted.

6.2.5.7 Lpspi_Ip_AsyncTransmitFast()

```
Lpspi_Ip_StatusType Lpspi_Ip_AsyncTransmitFast (
    const Lpspi_Ip_FastTransferType * FastTransferCfg,
    uint8 NumberOfTransfer,
    Lpspi_Ip_CallbackType EndCallback )
```

LPSPI asynchronous transmission fast.

This function initializes an asynchronous transmission for multiple transfers session and CPU used only for processing at the end of sequence transfer. The list of transfers session is composed of an array of fast transfers settings. The settings array is defined by the user needs: it contains entries parameters to be configured for each transfer session as defined in [Lpspi_Ip_FastTransferType](#).

How to use this interface:

1. Use the "Lpspi_Ip_FastTransferType" to create a list(array) of transfer session. Each field in [Lpspi_Ip_FastTransferType](#) for each transfer session must be configured. Note: This feature requires:
 - a. The parameters SpiBaudrate, SpiHwUnit, SpiTimeClk2Cs, SpiTimeCs2Clk, SpiTimeCs2Cs, SpiDataWidth, SpiTransferStart in all External Devices used(pointed by ExternalDevice) must be the same in each transfer session.
 - b. In each transfer section, the number of data buffer(Length) is NOT higher than 32767 if SpiDataWidth < 9.
 - c. Only Master mode is supported(SpiPhyUnit/SpiPhyUnitMode = SPI_MASTER).
 - d. Make sure that SpiPhyUnit/SpiMaxDmaFastTransfer value must NOT lower than total of transfer sessions.
 - e. Make sure that number of ScatterGathers configuration in SpiPhyUnit/SpiPhyTxDmaChannel must NOT lower than total of transfer sessions plus number of time request CS de-assert(KeepCs = FALSE) at the end of transfer session in the list configured.
 - f. Make sure that number of ScatterGathers configuration in each SpiPhyUnit/SpiPhyRxDmaChannel must NOT lower than total of transfer sessions.

2. Call the "Lpspi_Ip_AsyncTransmitFast()" interface.

Example: The user shall create the desired configuration list for his specific application. For example use case:

- Requiring 2 transfers session, keep CS assert at the end of first transfer session.
- Transfer session 1:
 - Use SpiExternalDevice_0 with SpiCsIdentifier = PCS0, SpiCsContinuous = TRUE.
 - Send 5 bytes. Tx buffer is "uint8 u8TxBuffer1[5u]={0,1,2,3,4};". Rx buffer is "uint8 u8RxBuffer1[5u];".
 - Keep CS assert at the end of this transfer session.
- Transfer session 2:
 - Use SpiExternalDevice_0 with SpiCsIdentifier = PCS0, SpiCsContinuous = TRUE.
 - Send 10 bytes with default transmit data value is 5. Tx buffer is NULL_PTR. Rx buffer is "uint8 u8RxBuffer2[10u];".
 - This is last transfer session, so CS will not kipped by default at the end of last transfer session.
- Configuration example on configuration tool:
 - SpiGeneral/SpiEnableDmaFastTransferSupport = true.
 - SpiPhyUnit/SpiMaxDmaFastTransfer = 2(2 transfers session).
 - Number of ScatterGathers configuration for SpiPhyTxDmaChannel is 3(2 transfers session + 1 time CS de-assert at the end of last transfer session).
 - Number of ScatterGathers configuration for SpiPhyRxDmaChannel is 2(2 transfers session).
- Call "UserCallbackFunc" when Fast transfer completed.
- Coding example: void UserCallbackFunc(uint8 Instance, Lpspi_Ip_EventType event); [Lpspi_Ip_FastTransferType](#)
 aUserFastTransferCfgList[2u] = { { Lpspi_Ip_DeviceAttributes_SpiExternalDevice_0_BOARD_Init←
 Peripherals, ->Point to External Device 0 configuration generated by configuration tool u8TxBuffer1, ->
 Store pointer for Tx buffer u8RxBuffer1, -> Store pointer for Rx buffer 0u, -> Default transmit data, don't
 care due to Tx buffer is not NULL_PTR 5u, -> Number of bytes to be sent (boolean)TRUE -> Keep CS signal
 at the end of this transfer session }, { Lpspi_Ip_DeviceAttributes_SpiExternalDevice_0_BOARD_Init←
 Peripherals, -> Point to external device configuration NULL_PTR, -> Store pointer for Tx buffer u8RxBuffer2,
 -> Store pointer for Rx buffer 5u, -> Default transmit data, don't care due to Tx buffer is not NULL_PTR
 10u, -> Number of bytes to be sent (boolean)FALSE -> Not keep CS signal at the end of this transfer session,
 don't care this parameter for last transfer } }; Lpspi_Ip_AsyncTransmitFast(aUserFastTransferCfgList, 2u,
 &UserCallbackFunc);

Parameters

	<i>[in-out]</i>	FastTransferCfg - pointer to the list of transfers section configuration.
in	<i>NumberOfTransfer</i>	- number of transfers session in the list is pointed by FastTransferCfg.
in	<i>EndCallback</i>	- callback function is called at the end of sequence transfer.

Returns

LPSPi_IP_STATUS_SUCCESS: Transmission command has been accepted. LPSPi_IP_STATUS_FAIL: Transmission command has not been accepted.

6.2.5.8 Lpspi_Ip_GetStatus()

```
Lpspi_Ip_HwStatusType Lpspi_Ip_GetStatus (
    uint8 Instance )
```

Get status of HW unit.

This function returns the status of the specified SPI Hardware microcontroller peripheral.

Parameters

in	<i>Instance</i>	- SPI peripheral instance number.
----	-----------------	-----------------------------------

Returns

Lpspi_Ip_HwStatusType

6.2.5.9 Lpspi_Ip_ManageBuffers()

```
void Lpspi_Ip_ManageBuffers (
    uint8 Instance )
```

Process transfer in POLLING mode.

This function shall poll the SPI interrupts linked to SPI peripheral instance allocated to the transmission of data to enable the evolution of transmission state machine.

Parameters

in	<i>Instance</i>	- SPI peripheral instance number.
----	-----------------	-----------------------------------

Returns

void

6.2.5.10 Lpspi_Ip_UpdateFrameSize()

```
Lpspi_Ip_StatusType Lpspi_Ip_UpdateFrameSize (
    const Lpspi_Ip_ExternalDeviceType * ExternalDevice,
    uint16 FrameSize )
```

LPSPI change frame size.

This function updates frame size of specific external device configuration for next transfers.

Parameters

in	<i>ExternalDevice</i>	- pointer to the external device configuration.
in	<i>FrameSize</i>	- Frame size. Supported values are: 8, 16, 32 and if > 32 it should be a pair number.

Returns

LPSPI_IP_STATUS_SUCCESS: Setting command has been accepted. LPSPI_IP_STATUS_FAIL: Setting command has not been accepted.

6.2.5.11 Lpspi_Ip_UpdateLsb()

```
Lpspi_Ip_StatusType Lpspi_Ip_UpdateLsb (
    const Lpspi_Ip_ExternalDeviceType * ExternalDevice,
    boolean Lsb )
```

LPSPI change bit order.

This function updates bits order LSB or MSB of specific external device configuration for next transfer.

Parameters

in	<i>ExternalDevice</i>	- pointer to the external device configuration.
in	<i>Lsb</i>	- Data is transferred LSB first or not.

Returns

LPSPI_IP_STATUS_SUCCESS: Setting command has been accepted. LPSPI_IP_STATUS_FAIL: Setting command has not been accepted.

6.2.5.12 Lpspi_Ip_UpdateDefaultTransmitData()

```
Lpspi_Ip_StatusType Lpspi_Ip_UpdateDefaultTransmitData (
    const Lpspi_Ip_ExternalDeviceType * ExternalDevice,
    uint32 DefaultData )
```

LPSPI change default transmit data.

This function updates default transmit data of specific external device configuration for next transfer.

Parameters

in	<i>ExternalDevice</i>	- pointer to the external device configuration.
in	<i>DefaultData</i>	- New default transmit data.

Returns

LPSPI_IP_STATUS_SUCCESS: Setting command has been accepted. LPSPI_IP_STATUS_FAIL: Setting command has not been accepted.

6.2.5.13 Lpspi_Ip_UpdateTransferMode()

```
Lpspi_Ip_StatusType Lpspi_Ip_UpdateTransferMode (
    uint8 Instance,
    Lpspi_Ip_ModeType Mode )
```

LPSPI change transfer mode.

This function updates the asynchronous mechanism mode for the specified SPI Hardware microcontroller peripheral.

Parameters

in	<i>Instance</i>	- SPI peripheral instance number.
in	<i>Mode</i>	- new mode (interrupt or polling).

Returns

LPSPI_IP_STATUS_SUCCESS: Setting command has been accepted. LPSPI_IP_STATUS_FAIL: Setting command has not been accepted.

6.2.5.14 Lpspi_Ip_Cancel()

```
void Lpspi_Ip_Cancel (
    uint8 Instance )
```

LPSPi cancel current asynchronous transmission.

This function cancels an asynchronous transmission in progress for the specified SPI Hardware microcontroller peripheral.

Parameters

in	<i>Instance</i>	- SPI peripheral instance number.
----	-----------------	-----------------------------------

Returns

void

6.2.6 Variable Documentation

6.2.6.1 Lpspi_Ip_DeviceAttributes_SpiExternalDevice_LPSPi0_VS_0

```
const Lpspi_Ip_ExternalDeviceType Lpspi_Ip_DeviceAttributes_SpiExternalDevice_LPSPi0_VS_0 [extern]
```

Export Post-Build configurations.

6.3 Spi Driver

6.3.1 Detailed Description

Data Structures

- struct [Spi_SequenceConfigType](#)
This structure contains all the needed data to configure one SPI Sequence. [More...](#)
- struct [Spi_SeqsConfigType](#)
This structure contains Sequence configuration. [More...](#)
- struct [Spi_SequenceStateType](#)
Internal structure used to manage the sequence state. [More...](#)
- struct [Spi_JobStateType](#)
Internal structure used to manage the job state. [More...](#)
- struct [Spi_JobConfigType](#)
This is the structure containing all the parameters needed to completely define a Job. [More...](#)
- struct [Spi_JobsCfgType](#)
This is the structure containing Job configuration. [More...](#)
- struct [Spi_BufferDescriptorType](#)
The structure contains the pointers to the Tx/Rx memory locations for the given buffer (IB or EB). [More...](#)
- struct [Spi_ChannelStateType](#)
Internal structure used to manage the channel state. [More...](#)
- struct [Spi_HWUnitQueue](#)
This structure holds the HWUnit scheduling queue. [More...](#)
- struct [Spi_ChannelConfigType](#)
The structure contains the channel configuration parameters. [More...](#)
- struct [Spi_ChannelsCfgType](#)
The structure contains the channel configuration. [More...](#)
- struct [Spi_ConfigType](#)
This is the top level structure containing all the needed parameters for the SPI Handler Driver. [More...](#)
- struct [Spi_Ipw_IpConfigType](#)
- struct [Spi_Ipw_ExternalDeviceType](#)
- struct [Spi_HWUnitConfigType](#)
This structure holds the HWUnit configuration parameters. [More...](#)
- struct [Spi_PhyUnitsConfigType](#)
This structure holds the PhyUnit configuration. [More...](#)
- struct [Spi_ExternalDeviceConfigType](#)
This structure holds the configuration parameters for each ExternalDevice. [More...](#)
- struct [Spi_ExDevicesConfigType](#)
This structure holds the ExternalDevice configuration. [More...](#)

Macros

- `#define SPI_PHYUNIT_SYNC_U32`
Define state of hardware unit for synchronous transmission.
- `#define SPI_E_PARAM_CHANNEL`
API service called with wrong parameter of Channel.
- `#define SPI_E_PARAM_JOB`
API service called with wrong parameter of Job.
- `#define SPI_E_PARAM_SEQ`
API service called with wrong parameter of Sequence.
- `#define SPI_E_PARAM_LENGTH`
API service called with wrong parameter of external buffer length.
- `#define SPI_E_PARAM_UNIT`
API service called with wrong parameter of HWUnit.
- `#define SPI_E_PARAM_CONFIG`
API service called with wrong resource assigned.
- `#define SPI_E_UNINIT`
API service used without module initialization.
- `#define SPI_E_SEQ_PENDING`
Services called in a wrong sequence.
- `#define SPI_E_SEQ_IN_PROCESS`
Synchronous transmission service called at wrong time.
- `#define SPI_E_ALREADY_INITIALIZED`
API SPI_Init service called while the SPI driver has already been initialized.
- `#define SPI_E_CONFIG_OUT_OF_RANGE`
The number of sequences, jobs or channels exceeds precompile time sizes.
- `#define SPI_E_INIT_FAILED`
API Spi_Init was called with wrong configuration pointer.
- `#define SPI_E_PARAM_EB_UNIT`
When a sequence contains uninitialized external buffers.
- `#define SPI_E_SEQ_EMPTY`
No job in sequence.
- `#define SPI_E_JOB_EMPTY`
No channel in job.
- `#define SPI_E_PARAM_POINTER`
If the parameter versioninfo or Spi configuration is NULL_PTR.
- `#define SPI_INIT_ID`
API service ID for SPI Init function.
- `#define SPI_DEINIT_ID`
API service ID for SPI DeInit function.
- `#define SPI_WRITEIB_ID`
API service ID for SPI write IB function.
- `#define SPI_ASYNCTRANSMIT_ID`
API service ID for SPI async transmit function.
- `#define SPI_READIB_ID`
API service ID for SPI read IB function.
- `#define SPI_SETUPEB_ID`

- API service ID for SPI setup EB function.*

• #define [SPI_GETSTATUS_ID](#)
- API service ID for SPI get status function.*

• #define [SPI_GETJOBRESULT_ID](#)
- API service ID for SPI get job result function.*

• #define [SPI_GETSEQUENCERESULT_ID](#)
- API service ID for SPI get sequence result function.*

• #define [SPI_GETVERSIONINFO_ID](#)
- API service ID for SPI get version info function.*

• #define [SPI_SYNCTRANSMIT_ID](#)
- API service ID for SPI sync transmit function.*

• #define [SPI_GETHWUNITSTATUS_ID](#)
- API service ID for SPI get hwunit status function.*

• #define [SPI_CANCEL_ID](#)
- API service ID for SPI cancel function.*

• #define [SPI_SETASYNCMODE_ID](#)
- API service ID for SPI set async mode function.*

• #define [SPI_MAINFUNCTION_HANDLING_ID](#)
- API service ID for SPI main function.*

• #define [SPI_SETHWUNITASYNCMODE_ID](#)
- API service ID for SPI set HW Unit async mode.*

• #define [SPI_SETCLOCKMODE_ID](#)
- API service ID for SPI Set Clock Mode.*

• #define [SPI_JOB_PRIORITY_LEVELS_COUNT](#)
- The number of allowed job priority levels (0..3).*

Types Reference

- typedef uint8 [Spi_DataBufferType](#)

Type of application data buffer elements.
- typedef uint16 [Spi_NumberOfDataType](#)

Type for defining the number of data elements of the type Spi_DataBufferType.
- typedef uint8 [Spi_ChannelType](#)

Specifies the identification (ID) for a Channel.
- typedef uint16 [Spi_JobType](#)

Specifies the identification (ID) for a Job.
- typedef uint8 [Spi_SequenceType](#)

Specifies the identification (ID) for a sequence of jobs.
- typedef uint8 [Spi_HWUnitType](#)

Specifies the ID for a SPI Hardware microcontroller peripheral unit.
- typedef uint8 [Spi_ExternalDeviceType](#)

Contains the ID of an external device.

Enum Reference

- enum [Spi_StatusType](#)
This type defines a range of specific status for SPI Driver.
- enum [Spi_JobResultType](#)
This type defines a range of specific Jobs status for SPI Driver.
- enum [Spi_SeqResultType](#)
This type defines a range of specific Sequences status for SPI Driver.
- enum [Spi_BufferType](#)
The enumeration containing the designated values for buffer types (internal or external).
- enum [Spi_AsyncModeType](#)
Specifies the asynchronous mechanism mode for SPI buses handled asynchronously in Level 2.
- enum [Spi_HalfDuplexModeType](#)
Half duplex mode.
- enum [Spi_Ipw_SupportedIpsType](#)
This enum contains all IPs which can integrate SPI functionalities.

Function Reference

- void [Spi_JobTransferFinished](#) (const [Spi_JobConfigType](#) *JobConfig, [Spi_JobResultType](#) JobResult)
This function is called after a Job has been executed.
- void [Spi_GetVersionInfo](#) (Std_VersionInfoType *versioninfo)
This function returns the version information for the SPI driver.
- void [Spi_Init](#) (const [Spi_ConfigType](#) *ConfigPtr)
This function initializes the SPI driver.
- Std_ReturnType [Spi_DeInit](#) (void)
This function de-initializes the SPI driver.
- Std_ReturnType [Spi_WriteIB](#) ([Spi_ChannelType](#) Channel, const [Spi_DataBufferType](#) *DataBufferPtr)
This function writes the given data into the buffer of a specific channel.
- Std_ReturnType [Spi_ReadIB](#) ([Spi_ChannelType](#) Channel, [Spi_DataBufferType](#) *DataBufferPointer)
This function reads the data from the buffer of a channel and puts at the memory location.
- Std_ReturnType [Spi_AsyncTransmit](#) ([Spi_SequenceType](#) Sequence)
This function triggers the asynchronous transmission for the given sequence.
- Std_ReturnType [Spi_SetupEB](#) ([Spi_ChannelType](#) Channel, const [Spi_DataBufferType](#) *SrcDataBufferPtr, [Spi_DataBufferType](#) *DesDataBufferPtr, [Spi_NumberOfDataType](#) Length)
This function setup an external buffer to be used by a specific channel.
- [Spi_StatusType](#) [Spi_GetStatus](#) (void)
This function returns the status of the SPI driver.
- [Spi_JobResultType](#) [Spi_GetJobResult](#) ([Spi_JobType](#) Job)
This function is used to request the status of a specific job.
- [Spi_SeqResultType](#) [Spi_GetSequenceResult](#) ([Spi_SequenceType](#) Sequence)
This function is used to request the status of a specific sequence.
- Std_ReturnType [Spi_SyncTransmit](#) ([Spi_SequenceType](#) Sequence)
This function is used for synchronous transmission of a given sequence.
- [Spi_StatusType](#) [Spi_GetHWUnitStatus](#) ([Spi_HWUnitType](#) HWUnit)
This function is used to request the status of a specific SPI peripheral unit.

- void [Spi_Cancel](#) ([Spi_SequenceType](#) Sequence)
This function is used to request the cancelation of the given sequence.
- Std_ReturnType [Spi_SetAsyncMode](#) ([Spi_AsyncModeType](#) Mode)
This function specifies the asynchronous mode for the SPI busses handled asynchronously.
- Std_ReturnType [Spi_SetHWUnitAsyncMode](#) ([Spi_HWUnitType](#) HWUnit, [Spi_AsyncModeType](#) AsyncMode)
This function specifies the asynchronous mode for a given HWUnit.
- void [Spi_MainFunction_Handling](#) (void)
This function shall asynchronously poll SPI interrupts and call ISR if appropriate.

Variables

- [Spi_JobStateType](#) [Spi_axSpiJobState](#) [(1u)]
Extern arrays contain the state of Sequences, Jobs and Channels.

6.3.2 Data Structure Documentation

6.3.2.1 struct Spi_SequenceConfigType

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

Definition at line 566 of file Spi.h.

Data Fields

- [Spi_JobType](#) NumJobs
Number of jobs in the sequence.
- uint32 [SpiCoreUse](#)
CoreID used.
- const [Spi_JobType](#) * [JobIndexList](#)
Job index list.
- [Spi_NotifyType](#) * [EndNotification](#)
Job notification handler.
- uint8 [Interruptible](#)
Boolean indicating if the Sequence is interruptible or not.
- boolean [EnableDmaFastTransfer](#)
Boolean indicating if the Sequence is transferred in Dma fast mode or not.

6.3.2.1.1 Field Documentation

6.3.2.1.1.1 NumJobs `Spi_JobType` NumJobs

Number of jobs in the sequence.

Definition at line 569 of file Spi.h.

6.3.2.1.1.2 SpiCoreUse `uint32` SpiCoreUse

CoreID used.

Definition at line 571 of file Spi.h.

6.3.2.1.1.3 JobIndexList `const Spi_JobType*` JobIndexList

Job index list.

Definition at line 573 of file Spi.h.

6.3.2.1.1.4 EndNotification `Spi_NotifyType*` EndNotification

Job notification handler.

Definition at line 575 of file Spi.h.

6.3.2.1.1.5 Interruptible `uint8` Interruptible

Boolean indicating if the Sequence is interruptible or not.

Definition at line 577 of file Spi.h.

6.3.2.1.1.6 EnableDmaFastTransfer `boolean` EnableDmaFastTransfer

Boolean indicating if the Sequence is transferred in Dma fast mode or not.

Definition at line 580 of file Spi.h.

6.3.2.2 struct Spi_SeqsConfigType

This structure contains Sequence configuration.

Definition at line 592 of file Spi.h.

Data Fields

Type	Name	Description
const Spi_SequenceConfigType *	SeqConfig	Point to Sequence configuration.

6.3.2.3 struct Spi_SequenceStateType

Internal structure used to manage the sequence state.

Definition at line 601 of file Spi.h.

Data Fields

Type	Name	Description
Spi_SeqResultType	Result	Sequence Result.
const Spi_SequenceConfigType *	Sequence	Pointer to the configuration.
const Spi_JobType *	CurrentJobIndexPointer	Position in JobIndexList to the job in transmission of an async sequence.
Spi_JobType	RemainingJobs	Number of jobs in a pending async sequence, not yet transmitted.

6.3.2.4 struct Spi_JobStateType

Internal structure used to manage the job state.

Definition at line 619 of file Spi.h.

Data Fields

Type	Name	Description
Spi_JobResultType	Result	Job Result.
Spi_SequenceStateType *	AsyncCrtSequenceState	Pointer to the state information of the async sequence.
Spi_JobType	AsyncNextJob	Pointer to the next async job planned for transmission.

6.3.2.5 struct Spi_JobConfigType

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

Definition at line 637 of file Spi.h.

Data Fields

- `Spi_ChannelType NumChannels`
Number of channels in the job.
- `const Spi_ChannelType * ChannelIndexList`
Channel index list.
- `Spi_NotifyType * EndNotification`
Job end notification.
- `Spi_NotifyType * StartNotification`
Job start notification.
- `sint8 Priority`
Priority.
- `uint32 SpiCoreUse`
CoreID used.
- `Spi_JobStateType * JobState`
Implementation specific field referencing the channel internal state.
- `Spi_HWUnitType HWUnit`
HWUnit.
- `Spi_ExternalDeviceType ExternalDevice`
ExternalDevice.
- `const Spi_ExDevicesConfigType * ExternalDeviceConfig`
Implementation specific field: cached LLD device attributes.

6.3.2.5.1 Field Documentation

6.3.2.5.1.1 NumChannels `Spi_ChannelType NumChannels`

Number of channels in the job.

Definition at line 640 of file Spi.h.

6.3.2.5.1.2 ChannelIndexList `const Spi_ChannelType* ChannelIndexList`

Channel index list.

Definition at line 642 of file Spi.h.

6.3.2.5.1.3 EndNotification `Spi_NotifyType* EndNotification`

Job end notification.

Definition at line 644 of file Spi.h.

6.3.2.5.1.4 StartNotification `Spi_NotifyType* StartNotification`

Job start notification.

Definition at line 646 of file Spi.h.

6.3.2.5.1.5 Priority `sint8 Priority`

Priority.

Definition at line 648 of file Spi.h.

6.3.2.5.1.6 SpiCoreUse `uint32 SpiCoreUse`

CoreID used.

Definition at line 650 of file Spi.h.

6.3.2.5.1.7 JobState `Spi_JobStateType* JobState`

Implementation specific field referencing the channel internal state.

Definition at line 652 of file Spi.h.

6.3.2.5.1.8 HWUnit `Spi_HWUnitType HWUnit`

HWUnit.

Definition at line 654 of file Spi.h.

6.3.2.5.1.9 ExternalDevice `Spi_ExternalDeviceType ExternalDevice`

ExternalDevice.

Definition at line 656 of file Spi.h.

6.3.2.5.1.10 ExternalDeviceConfig `const Spi_ExDevicesConfigType* ExternalDeviceConfig`

Implementation specific field: cached LLD device attributes.

Definition at line 658 of file Spi.h.

6.3.2.6 struct Spi_JobsCfgType

This is the structure containing Job configuration.

Definition at line 664 of file Spi.h.

Data Fields

Type	Name	Description
const Spi_JobConfigType *	JobCfg	Point to Job configuration.

6.3.2.7 struct Spi_BufferDescriptorType

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

Definition at line 674 of file Spi.h.

Data Fields

Type	Name	Description
const Spi_DataBufferType *	ExternalBufferTX	Transmit buffer pointer.
Spi_DataBufferType *	InternalBufferTX	
Spi_DataBufferType *	BufferRX	Receive buffer pointer.

6.3.2.8 struct Spi_ChannelStateType

Internal structure used to manage the channel state.

Definition at line 687 of file Spi.h.

Data Fields

Type	Name	Description
uint8	Flags	Default Transmit Enabled.
Spi_NumberOfDataType	Length	Actual Transfer size for EB.

6.3.2.9 struct Spi_HWUnitQueue

This structure holds the HWUnit scheduling queue.

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

Definition at line 699 of file Spi.h.

Data Fields

Type	Name	Description
Spi_JobType	ScheduledJobsListHead[(4)]	Array of the IDs of jobs to be scheduled, for each priority level.

Data Fields

Type	Name	Description
Spi_JobType	ScheduledJobsListTail[(4)]	Array of the IDs of last jobs in queues, for each priority level.
sint8	MaxScheduledPriority	Array of the IDs of last jobs in queues, for each priority level.
Spi_StatusType	Status	DSPI state.
Spi_ChannelType	Channel	Current channel index in Job
Spi_JobType	Job	Current job index

6.3.2.10 struct Spi_ChannelConfigType

The structure contains the channel configuration parameters.

Definition at line 717 of file Spi.h.

Data Fields

Type	Name	Description
Spi_BufferType	BufferType	Buffer Type IB/EB.
uint8	FrameSize	Data frame size.
boolean	Lsb	Bite order (MSB/LSB).
Spi_HalfDuplexModeType	HalfDuplexMode	Half duplex mode.
uint32	DefaultTransmitValue	Default Transmit Value.
Spi_NumberOfDataType	Length	Data length.
Spi_BufferDescriptorType *	BufferDescriptor	Buffer Descriptor.
uint32	SpiCoreUse	CoreID assigned.
Spi_ChannelStateType *	ChannelState	Implementation specific field referencing the channel internal state.

6.3.2.11 struct Spi_ChannelsCfgType

The structure contains the channel configuration.

Definition at line 746 of file Spi.h.

Data Fields

Type	Name	Description
const Spi_ChannelConfigType *	ChannelCfg	Point to Channel configuration.

6.3.2.12 struct Spi_ConfigType

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

Definition at line 766 of file Spi.h.

Data Fields

- uint16 [MaxExternalDevice](#)
Number of external devices defined in the configuration.
- [Spi_ChannelType](#) [SpiMaxChannel](#)
Number of channels defined in the configuration.
- [Spi_JobType](#) [SpiMaxJob](#)
Number of jobs defined in the configuration.
- [Spi_SequenceType](#) [SpiMaxSequence](#)
Number of sequences defined in the configuration.
- uint32 [SpiCoreUse](#)
CoreID used.
- const [Spi_ChannelsCfgType](#) * [ChannelConfig](#)
Pointer to Array of channels defined in the configuration.
- const [Spi_JobsCfgType](#) * [JobConfig](#)
Pointer to Array of jobs defined in the configuration.
- const [Spi_SeqsConfigType](#) * [SequenceConfig](#)
Pointer to Array of sequences defined in the configuration.
- const [Spi_ExDevicesConfigType](#) * [ExternalDeviceConfig](#)
External device unit attributes.
- const [Spi_PhyUnitsConfigType](#) * [HWUnitConfig](#)
Pointer to Array of LLD DSPI device instances.
- const Mcal_DemErrorType [SpiErrorHardwareCfg](#)
SPI Driver DEM Error: SPI_E_HARDWARE_ERROR.

6.3.2.12.1 Field Documentation

6.3.2.12.1.1 MaxExternalDevice `uint16 MaxExternalDevice`

Number of external devices defined in the configuration.

Definition at line 769 of file Spi.h.

6.3.2.12.1.2 SpiMaxChannel `Spi_ChannelType SpiMaxChannel`

Number of channels defined in the configuration.

Definition at line 771 of file Spi.h.

6.3.2.12.1.3 SpiMaxJob `Spi_JobType` SpiMaxJob

Number of jobs defined in the configuration.

Definition at line 773 of file Spi.h.

6.3.2.12.1.4 SpiMaxSequence `Spi_SequenceType` SpiMaxSequence

Number of sequences defined in the configuration.

Definition at line 775 of file Spi.h.

6.3.2.12.1.5 SpiCoreUse `uint32` SpiCoreUse

CoreID used.

Definition at line 777 of file Spi.h.

6.3.2.12.1.6 ChannelConfig `const Spi_ChannelsCfgType*` ChannelConfig

Pointer to Array of channels defined in the configuration.

Definition at line 779 of file Spi.h.

6.3.2.12.1.7 JobConfig `const Spi_JobsCfgType*` JobConfig

Pointer to Array of jobs defined in the configuration.

Definition at line 781 of file Spi.h.

6.3.2.12.1.8 SequenceConfig `const Spi_SeqsConfigType*` SequenceConfig

Pointer to Array of sequences defined in the configuration.

Definition at line 783 of file Spi.h.

6.3.2.12.1.9 ExternalDeviceConfig `const Spi_ExDevicesConfigType* ExternalDeviceConfig`

External device unit attributes.

Definition at line 785 of file Spi.h.

6.3.2.12.1.10 HWUnitConfig `const Spi_PhyUnitsConfigType* HWUnitConfig`

Pointer to Array of LLD DSPI device instances.

Definition at line 787 of file Spi.h.

6.3.2.12.1.11 SpiErrorHardwareCfg `const Mcal_DemErrorType SpiErrorHardwareCfg`

SPI Driver DEM Error: SPI_E_HARDWARE_ERROR.

Definition at line 790 of file Spi.h.

6.3.2.13 struct Spi_Ipw_IpConfigType

@bried This union contains config structure for all IPs available.

Definition at line 146 of file Spi_Ipw_Types.h.

6.3.2.14 struct Spi_Ipw_ExternalDeviceType

@bried This union contains config structure for all external device available.

Definition at line 162 of file Spi_Ipw_Types.h.

6.3.2.15 struct Spi_HWUnitConfigType

This structure holds the HWUnit configuration parameters.

Definition at line 175 of file Spi_Ipw_Types.h.

6.3.2.16 struct Spi_PhyUnitsConfigType

This structure holds the PhyUnit configuration.

Definition at line 187 of file Spi_Ipw_Types.h.

Data Fields

Type	Name	Description
const Spi_HWUnitConfigType *	PhyUnitConfig	Point to PhyUnit configuration.

6.3.2.17 struct Spi_ExternalDeviceConfigType

This structure holds the configuration parameters for each ExternalDevice.

Definition at line 196 of file Spi_Ipw_Types.h.

6.3.2.18 struct Spi_ExDevicesConfigType

This structure holds the ExternalDevice configuration.

Definition at line 207 of file Spi_Ipw_Types.h.

Data Fields

Type	Name	Description
const Spi_ExternalDeviceConfigType *	ExDeviceConfig	Point to ExternalDevice configuration.

6.3.3 Macro Definition Documentation

6.3.3.1 SPI_PHYUNIT_SYNC_U32

```
#define SPI_PHYUNIT_SYNC_U32
```

Define state of hardware unit for synchronous transmission.

Definition at line 128 of file Spi.h.

6.3.3.2 SPI_E_PARAM_CHANNEL

```
#define SPI_E_PARAM_CHANNEL
```

API service called with wrong parameter of Channel.

Definition at line 136 of file Spi.h.

6.3.3.3 SPI_E_PARAM_JOB

```
#define SPI_E_PARAM_JOB
```

API service called with wrong parameter of Job.

Definition at line 142 of file Spi.h.

6.3.3.4 SPI_E_PARAM_SEQ

```
#define SPI_E_PARAM_SEQ
```

API service called with wrong parameter of Sequence.

Definition at line 148 of file Spi.h.

6.3.3.5 SPI_E_PARAM_LENGTH

```
#define SPI_E_PARAM_LENGTH
```

API service called with wrong parameter of external buffer length.

Definition at line 154 of file Spi.h.

6.3.3.6 SPI_E_PARAM_UNIT

```
#define SPI_E_PARAM_UNIT
```

API service called with wrong parameter of HWUnit.

Definition at line 160 of file Spi.h.

6.3.3.7 SPI_E_PARAM_CONFIG

```
#define SPI_E_PARAM_CONFIG
```

API service called with wrong resource assigned.

Definition at line 166 of file Spi.h.

6.3.3.8 SPI_E_UNINIT

```
#define SPI_E_UNINIT
```

API service used without module initialization.

Definition at line 172 of file Spi.h.

6.3.3.9 SPI_E_SEQ_PENDING

```
#define SPI_E_SEQ_PENDING
```

Services called in a wrong sequence.

Definition at line 178 of file Spi.h.

6.3.3.10 SPI_E_SEQ_IN_PROCESS

```
#define SPI_E_SEQ_IN_PROCESS
```

Synchronous transmission service called at wrong time.

Definition at line 184 of file Spi.h.

6.3.3.11 SPI_E_ALREADY_INITIALIZED

```
#define SPI_E_ALREADY_INITIALIZED
```

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

Definition at line 190 of file Spi.h.

6.3.3.12 SPI_E_CONFIG_OUT_OF_RANGE

```
#define SPI_E_CONFIG_OUT_OF_RANGE
```

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

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

Definition at line 200 of file Spi.h.

6.3.3.13 SPI_E_INIT_FAILED

```
#define SPI_E_INIT_FAILED
```

API Spi_Init was called with wrong configuration pointer.

Definition at line 207 of file Spi.h.

6.3.3.14 SPI_E_PARAM_EB_UNIT

```
#define SPI_E_PARAM_EB_UNIT
```

When a sequence contains uninitialized external buffers.

Definition at line 214 of file Spi.h.

6.3.3.15 SPI_E_SEQ_EMPTY

```
#define SPI_E_SEQ_EMPTY
```

No job in sequence.

Definition at line 221 of file Spi.h.

6.3.3.16 SPI_E_JOB_EMPTY

```
#define SPI_E_JOB_EMPTY
```

No channel in job.

Definition at line 228 of file Spi.h.

6.3.3.17 SPI_E_PARAM_POINTER

```
#define SPI_E_PARAM_POINTER
```

If the parameter versioninfo or Spi configuration is NULL_PTR.

Definition at line 234 of file Spi.h.

6.3.3.18 SPI_INIT_ID

```
#define SPI_INIT_ID
```

API service ID for SPI Init function.

Parameters used when raising an error or exception.

Definition at line 251 of file Spi.h.

6.3.3.19 SPI_DEINIT_ID

```
#define SPI_DEINIT_ID
```

API service ID for SPI DeInit function.

Parameters used when raising an error or exception.

Definition at line 258 of file Spi.h.

6.3.3.20 SPI_WRITEIB_ID

```
#define SPI_WRITEIB_ID
```

API service ID for SPI write IB function.

Parameters used when raising an error or exception.

Definition at line 265 of file Spi.h.

6.3.3.21 SPI_ASYNCTRANSMIT_ID

```
#define SPI_ASYNCTRANSMIT_ID
```

API service ID for SPI async transmit function.

Parameters used when raising an error or exception.

Definition at line 272 of file Spi.h.

6.3.3.22 SPI_READIB_ID

```
#define SPI_READIB_ID
```

API service ID for SPI read IB function.

Parameters used when raising an error or exception.

Definition at line 279 of file Spi.h.

6.3.3.23 SPI_SETUPEB_ID

```
#define SPI_SETUPEB_ID
```

API service ID for SPI setup EB function.

Parameters used when raising an error or exception.

Definition at line 286 of file Spi.h.

6.3.3.24 SPI_GETSTATUS_ID

```
#define SPI_GETSTATUS_ID
```

API service ID for SPI get status function.

Parameters used when raising an error or exception.

Definition at line 293 of file Spi.h.

6.3.3.25 SPI_GETJOBRESULT_ID

```
#define SPI_GETJOBRESULT_ID
```

API service ID for SPI get job result function.

Parameters used when raising an error or exception.

Definition at line 300 of file Spi.h.

6.3.3.26 SPI_GETSEQUENCERESULT_ID

```
#define SPI_GETSEQUENCERESULT_ID
```

API service ID for SPI get sequence result function.

Parameters used when raising an error or exception.

Definition at line 307 of file Spi.h.

6.3.3.27 SPI_GETVERSIONINFO_ID

```
#define SPI_GETVERSIONINFO_ID
```

API service ID for SPI get version info function.

Parameters used when raising an error or exception.

Definition at line 314 of file Spi.h.

6.3.3.28 SPI_SYNCTRANSMIT_ID

```
#define SPI_SYNCTRANSMIT_ID
```

API service ID for SPI sync transmit function.

Parameters used when raising an error or exception.

Definition at line 321 of file Spi.h.

6.3.3.29 SPI_GETHWUNITSTATUS_ID

```
#define SPI_GETHWUNITSTATUS_ID
```

API service ID for SPI get hwunit status function.

Parameters used when raising an error or exception.

Definition at line 328 of file Spi.h.

6.3.3.30 SPI_CANCEL_ID

```
#define SPI_CANCEL_ID
```

API service ID for SPI cancel function.

Parameters used when raising an error or exception.

Definition at line 335 of file Spi.h.

6.3.3.31 SPI_SETASYNCMODE_ID

```
#define SPI_SETASYNCMODE_ID
```

API service ID for SPI set async mode function.

Parameters used when raising an error or exception.

Definition at line 342 of file Spi.h.

6.3.3.32 SPI_MAINFUNCTION_HANDLING_ID

```
#define SPI_MAINFUNCTION_HANDLING_ID
```

API service ID for SPI main function.

Parameters used when raising an error or exception

Definition at line 349 of file Spi.h.

6.3.3.33 SPI_SETHWUNITASYNCMODE_ID

```
#define SPI_SETHWUNITASYNCMODE_ID
```

API service ID for SPI set HW Unit async mode.

Parameters used when raising an error or exception.

Definition at line 356 of file Spi.h.

6.3.3.34 SPI_SETCLOCKMODE_ID

```
#define SPI_SETCLOCKMODE_ID
```

API service ID for SPI Set Clock Mode.

Parameters used when raising an error or exception.

Definition at line 363 of file Spi.h.

6.3.3.35 SPI_JOB_PRIORITY_LEVELS_COUNT

```
#define SPI_JOB_PRIORITY_LEVELS_COUNT
```

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

The Priority has to be sint8.

Definition at line 371 of file Spi.h.

6.3.4 Types Reference

6.3.4.1 Spi_DataBufferType

```
typedef uint8 Spi_DataBufferType
```

Type of application data buffer elements.

Definition at line 485 of file Spi.h.

6.3.4.2 Spi_NumberOfDataType

```
typedef uint16 Spi_NumberOfDataType
```

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

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

Definition at line 494 of file Spi.h.

6.3.4.3 Spi_ChannelType

```
typedef uint8 Spi_ChannelType
```

Specifies the identification (ID) for a Channel.

Definition at line 525 of file Spi.h.

6.3.4.4 Spi_JobType

```
typedef uint16 Spi_JobType
```

Specifies the identification (ID) for a Job.

Definition at line 532 of file Spi.h.

6.3.4.5 Spi_SequenceType

```
typedef uint8 Spi_SequenceType
```

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

Definition at line 539 of file Spi.h.

6.3.4.6 Spi_HWUnitType

```
typedef uint8 Spi_HWUnitType
```

Specifies the ID for a SPI Hardware microcontroller peripheral unit.

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

Definition at line 549 of file Spi.h.

6.3.4.7 Spi_ExternalDeviceType

```
typedef uint8 Spi_ExternalDeviceType
```

Contains the ID of an external device.

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

Definition at line 557 of file Spi.h.

6.3.5 Enum Reference

6.3.5.1 Spi_StatusType

```
enum Spi_StatusType
```

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

Enumerator

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

Definition at line 382 of file Spi.h.

6.3.5.2 Spi_JobResultType

```
enum Spi_JobResultType
```

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

Enumerator

SPI_JOB_OK	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.

Definition at line 394 of file Spi.h.

6.3.5.3 Spi_SeqResultType

```
enum Spi_SeqResultType
```

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

Enumerator

SPI_SEQ_OK	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.

Definition at line 408 of file Spi.h.

6.3.5.4 Spi_BufferType

```
enum Spi_BufferType
```

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

Enumerator

IB	The Channel is configured using Internal Buffer.
EB	The Channel is configured using External Buffer.

Definition at line 420 of file Spi.h.

6.3.5.5 Spi_AsyncModeType

enum `Spi_AsyncModeType`

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

`#if (SPI_LEVEL2 == SPI_LEVEL_DELIVERED)` Specifies the asynchronous mechanism mode for SPI buses 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.

Enumerator

<code>SPI_POLLING_MODE</code>	The asynchronous mechanism is ensured by polling, so interrupts related to SPI buses handled asynchronously are disabled.
<code>SPI_INTERRUPT_MODE</code>	The asynchronous mechanism is ensured by interrupt, so interrupts related to SPI buses handled asynchronously are enabled.

Definition at line 436 of file Spi.h.

6.3.5.6 Spi_HalfDuplexModeType

enum `Spi_HalfDuplexModeType`

Half duplex mode.

Enumerator

<code>SPI_HALF_DUPLEX_TRANSMIT</code>	Transmit only.
<code>SPI_HALF_DUPLEX_RECEIVE</code>	Receive only.
<code>SPI_FULL_DUPLEX</code>	Full duplex mode.

Definition at line 464 of file Spi.h.

6.3.5.7 Spi_Ipw_SupportedIpsType

enum `Spi_Ipw_SupportedIpsType`

This enum contains all IPs which can integrate SPI functionalities.

Definition at line 135 of file `Spi_Ipw_Types.h`.

6.3.6 Function Reference

6.3.6.1 Spi_JobTransferFinished()

```
void Spi_JobTransferFinished (
    const Spi_JobConfigType * JobConfig,
    Spi_JobResultType JobResult )
```

This function is called after a Job has been executed.

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

Parameters

in	<i>JobConfig</i>	The just transmited job pointer.
----	------------------	----------------------------------

Returns

void

Precondition

Pre-compile parameter `SPI_LEVEL_DELIVERED` shall be `SPI_LEVEL1` or `SPI_LEVEL2`.

6.3.6.2 Spi_GetVersionInfo()

```
void Spi_GetVersionInfo (
    Std_VersionInfoType * versioninfo )
```

This function returns the version information for the SPI driver.

This function returns the version information for the SPI driver.

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

Parameters

in, out	<i>VersionInfo</i>	Pointer to where to store the version information of this module.
---------	--------------------	---

Precondition

Pre-compile parameter `SPI_VERSION_INFO_API` shall be `STD_ON`.

Returns

void

6.3.6.3 Spi_Init()

```
void Spi_Init (
    const Spi_ConfigType * ConfigPtr )
```

This function initializes the SPI driver.

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

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

Parameters

in	<i>ConfigPtr</i>	Specifies the pointer to the configuration set
----	------------------	--

Returns

void

6.3.6.4 Spi_DeInit()

```
Std_ReturnType Spi_DeInit (
    void )
```

This function de-initializes the SPI driver.

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

Module Documentation

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

Returns

Std_ReturnType

Return values

<i>E_OK</i>	de-initialisation command has been accepted
<i>E_NOT_OK</i>	de-initialisation command has not been accepted

Precondition

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.

6.3.6.5 Spi_WriteIB()

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

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

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

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

Parameters

in	<i>Channel</i>	Channel ID
in	<i>DataBufferPtr</i>	Pointer to source data buffer

Returns

Std_ReturnType

Return values

<i>E_OK</i>	Command has been accepted
<i>E_NOT_OK</i>	Command has not been accepted

Precondition

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 `SPI_USAGE0` or `SPI_USAGE2`.

6.3.6.6 Spi_ReadIB()

```
Std_ReturnType Spi_ReadIB (
    Spi_ChannelType Channel,
    Spi_DataBufferType * DataBufferPointer )
```

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

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

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

Parameters

in	<i>Channel</i>	Channel ID
in, out	<i>DataBufferPointer</i>	Pointer to the memory location that will be written with the data in the internal buffer

Returns

Std_ReturnType

Return values

<i>E_OK</i>	read command has been accepted
<i>E_NOT_OK</i>	read command has not been accepted

Precondition

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 `SPI_USAGE0` or `SPI_USAGE2`.

6.3.6.7 Spi_AsyncTransmit()

```
Std_ReturnType Spi_AsyncTransmit (  
    Spi_SequenceType Sequence )
```

This function triggers the asynchronous transmission for the given sequence.

This function triggers the asynchronous transmission for the given sequence.

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

Parameters

in	<i>Sequence</i>	Sequence ID
----	-----------------	-------------

Returns

Std_ReturnType

Return values

<i>E_OK</i>	Transmission command has been accepted
<i>E_NOT_OK</i>	Transmission command has not been accepted

Precondition

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 `SPI_LEVEL1` or `SPI_LEVEL2`.

6.3.6.8 Spi_SetupEB()

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

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

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

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

Parameters

in	<i>Channel</i>	Channel ID
in	<i>SrcDataBufferPtr</i>	Pointer to the memory location that will hold the transmitted data
in	<i>Length</i>	Length of the data in the external buffer
out	<i>DesDataBufferPtr</i>	Pointer to the memory location that will hold the received data

Returns

Std_ReturnType

Return values

<i>E_OK</i>	Setup command has been accepted
<i>E_NOT_OK</i>	Setup command has not been accepted

Precondition

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 SPI_USAGE1 or SPI_USAGE2.

6.3.6.9 Spi_GetStatus()

```
Spi_StatusType Spi_GetStatus (
    void )
```

This function returns the status of the SPI driver.

This function returns the status of the SPI driver.

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

Returns

`Spi_StatusType`

Return values

<i>SPI_UNINIT</i>	The driver is un-initialized
<i>SPI_IDLE</i>	The driver has no pending transfers
<i>SPI_BUSY</i>	The driver is busy

Precondition

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`.

6.3.6.10 Spi_GetJobResult()

```
Spi_JobResultType Spi_GetJobResult (  
    Spi_JobType Job )
```

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

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

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

Parameters

in	<i>Job</i>	Job ID
----	------------	--------

Returns

`Spi_JobResultType`

Return values

<i>SPI_JOB_OK</i>	The job ended successfully
<i>SPI_JOB_PENDING</i>	The job is pending
<i>SPI_JOB_FAILED</i>	The job has failed

Precondition

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`.

6.3.6.11 Spi_GetSequenceResult()

```
Spi_SeqResultType Spi_GetSequenceResult (
    Spi_SequenceType Sequence )
```

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

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

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

Parameters

in	<i>Sequence</i>	Sequence ID
----	-----------------	-------------

Returns

`Spi_SeqResultType`

Return values

<i>SPI_SEQ_OK</i>	The sequence ended successfully
<i>SPI_SEQ_PENDING</i>	The sequence is pending
<i>SPI_SEQ_FAILED</i>	The sequence has failed

Precondition

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.

6.3.6.12 Spi_SyncTransmit()

```
Std_ReturnType Spi_SyncTransmit (
    Spi_SequenceType Sequence )
```

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

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

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

Parameters

in	<i>Sequence</i>	Sequence ID
----	-----------------	-------------

Returns

Std_ReturnType

Return values

<i>E_OK</i>	Transmission command has been completed successfully
<i>E_NOT_OK</i>	Transmission command has not been accepted

Precondition

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 SPI_LEVEL0 or SPI_LEVEL2

6.3.6.13 Spi_GetHWUnitStatus()

```
Spi_StatusType Spi_GetHWUnitStatus (
    Spi_HWUnitType HWUnit )
```

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

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

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

Parameters

in	<i>HWUnit</i>	The HW peripheral for which we need the status
----	---------------	--

Returns

Spi_StatusType

Return values

<i>SPI_UNINIT</i>	The peripheral is un-initialized
<i>SPI_IDLE</i>	The peripheral is in idle state
<i>SPI_BUSY</i>	The peripheral is busy

Precondition

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

6.3.6.14 Spi_Cancel()

```
void Spi_Cancel (
    Spi_SequenceType Sequence )
```

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

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

Module Documentation

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

Parameters

in	<i>Sequence</i>	Sequence ID
----	-----------------	-------------

Precondition

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

Postcondition

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

6.3.6.15 Spi_SetAsyncMode()

```
Std_ReturnType Spi_SetAsyncMode (  
    Spi_AsyncModeType Mode )
```

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

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

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

Parameters

in	<i>Mode</i>	This parameter specifies the asynchronous operating mode (SPI_POLLING_MODE or SPI_INTERRUPT_MODE)
----	-------------	---

Returns

Std_ReturnType

Return values

<i>E_OK</i>	The command ended successfully
<i>E_NOT_OK</i>	The command has failed

Precondition

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 `SPI_LEVEL2`

6.3.6.16 Spi_SetHWUnitAsyncMode()

```
Std_ReturnType Spi_SetHWUnitAsyncMode (
    Spi_HWUnitType HWUnit,
    Spi_AsyncModeType AsyncMode )
```

This function specifies the asynchronous mode for a given HWUnit.

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)

Parameters

in	<i>HWUnit</i>	The ID of the HWUnit to be configured
in	<i>AsyncMode</i>	This parameter specifies the asynchronous operating mode (<code>SPI_POLLING_MODE</code> or <code>SPI_INTERRUPT_MODE</code>)

Returns

Std_ReturnType

Return values

<i>E_OK</i>	The command ended successfully
<i>E_NOT_OK</i>	The command has failed

Precondition

Pre-compile parameter SPI_LEVEL_DELIVERED shall be SPI_LEVEL2 and SPI_HWUNIT_ASYNC_MODE should be on STD_ON

6.3.6.17 Spi_MainFunction_Handling()

```
void Spi_MainFunction_Handling (  
    void )
```

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

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

- Service ID: 0x10

Precondition

Pre-compile parameter SPI_LEVEL_DELIVERED shall be SPI_LEVEL1 or SPI_LEVEL2.

6.3.7 Variable Documentation

6.3.7.1 Spi_axSpiJobState

```
Spi_JobStateType Spi_axSpiJobState[(1u)] [extern]
```

Extern arrays contain the state of Sequences, Jobs and Channels.

How to Reach Us:

Home Page:

nxp.com

Web Support:

nxp.com/support

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

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

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

© 2023 NXP B.V.

