Integration Manual

for S32K14X SPI Driver

Document Number: IM2SPIASR4.2 Rev0002R1.0.2
Rev. 1.0

Contents

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

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0	26/04/2019	NXP MCAL Team	Updated version for ASR 4.2.2S32K14XR1.0.2

Chapter 2
Introduction

This integration manual describes the integration requirements for SPI Driver for S32K14X microcontrollers.

2.1 Supported Derivatives

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

Table 2-1. S32K14X Derivatives

NXP Semiconductors	s32k148_lqfp144, s32k148_lqfp176, s32k148_mapbga100, s32k146_lqfp144, s32k146_lqfp100, s32k146_lqfp64, s32k146_mapbga100, s32k144_lqfp100, s32k144_lqfp64, s32k144_mapbga100, s32k142_lqfp100, s32k142_lqfp64, s32k118_lqfp48, s32k118_lqfp64, s32k142_lqfp48, s32k144_lqfp48, s32k148_lqfp100

All of the above microcontroller devices are collectively named as S32K14X .

2.2 Overview

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

AUTOSAR

- paves the way for innovative electronic systems that further improve performance, safety and environmental friendliness.

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

2.3 About this Manual

This Technical Reference employs the following typographical conventions:

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

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

Notes and warnings are shown as below:

Note

This is a note.

2.4 Acronyms and Definitions

Table 2-2. Acronyms and Definitions

Term	Definition
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSMI	Basic Software Make file Interface
CS	Chip Select
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
FIFO	First In First Out
MIDE	Multi Integrated Development Environment
MCU	Micro Controller Unit
LSB	Least Significant Bit
MSB	Most Significant Bit
RAM	Random Access Memory

Table continues on the next page...

Table 2-2. Acronyms and Definitions (continued)

Term	Definition
SIU	Systems Integration Unit
SPI	Serial Peripheral Interface
SWS	Software Specification
VLE	Variable Length Encoding
XML	Extensible Markup Language
BSW	Basic Software
N/A	Not Applicable
ISR	Interrupt Service Routine
OS	Operating System
MCU	Microcontroller Unit
GUI	Graphical User Interface
PB Variant	Post Build Variant
PC Variant	Pre Compile Variant
LT Variant	Link Time Variant

2.5 Reference List

Table 2-3. Reference List

#	Title	Version
1	Specification of SPI Driver	AUTOSAR Release 4.2.2
2	S32K14X Reference Manual	Reference Manual, Rev. 9, 9/2018
3	S32K142 Mask Set Errata for Mask 0N33V (0N33V)	30/11/2017
4	S32K144 Mask Set Errata for Mask 0N57U (0N57U)	30/11/2017
5	S32K146 Mask Set Errata for Mask 0N73V (0N73V)	30/11/2017
6	S32K148 Mask Set Errata for Mask 0N20V (0N20V)	25/10/2018
7	S32K118 Mask Set Errata for Mask 0N97V (0N97V)	07/01/2019

Chapter 3
Building the Driver

This section describes the source files and various compilers, linker options used for building the Autosar SPI driver for NXP SemiconductorsS32K14X . It also explains the EB Tresos Studio plugin setup procedure.

3.1 Build Options

The SPI driver files are compiled using

- Green Hills Multi 7.1.4 / Compiler 2017.1.4
- (Linaro GCC 6.3-2017.06~dev) 6.3.1 20170509 (Wed Jan 24 16:21:45 CST 2018 build.sh rev=g27a1317 s=L631 Earmv7 -V release_g27a1317_build_Fed_Earmv7)
- IAR: V8.11.2

The compiler, linker flags used for building the driver are explained below:

Note

The TS_T40D2M10I2R0 plugin name is composed as follow:

TS_T = Target_Id

D = Derivative_Id

M = SW_Version_Major

I = SW_Version_Minor

R = Revision

(i.e. Target_Id = 40 identifies CORTEXM architecture and Derivative_Id = 2 identifies the S32K14X)

3.1.1 GHS Compiler/Linker/Assembler Options

Table 3-1. Compiler Options

Option	Description
-cpu=cortexm4	Selects target processor: Arm Cortex M4
-cpu=cortexm0plus	Selects target processor: Arm Cortex M0+
-ansi	Specifies ANSI C with extensions. This mode extends the ANSI X3.159-1989 standard with certain useful and compatible constructs.
-Osize	Optimize for size.
-dual_debug	Enables the generation of DWARF, COFF, or BSD debugging information in the object file
-G	Generates source level debugging information and allows procedure call from debugger's command line.
--no_exceptions	Disables support for exception handling
-Wundef	Generates warnings for undefined symbols in preprocessor expressions
-Wimplicit-int	Issues a warning if the return type of a function is not declared before it is called
-Wshadow	Issues a warning if the declaration of a local variable shadows the declaration of a variable of the same name declared at the global scope, or at an outer scope
-Wtrigraphs	Issues a warning for any use of trigraphs
-Wall	Enables all the warnings about constructions that some users consider questionable, and that are easy to avoid even in conjunction with macros.
--prototype_errors	Generates errors when functions referenced or called have no prototype
--incorrect_pragma_warnings	Valid #pragma directives with wrong syntax are treated as warnings
-noslashcomment	C++ like comments will generate a compilation error
-preprocess_assembly_files	Preprocesses assembly files
-nostartfile	Do not use Start files
--short_enum	Store enumerations in the smallest possible type
-c	Produces an object file (called input-file.o) for each source file.
--no_commons	Allocates uninitialized global variables to a section and initializes them to zero at program startup.
-keeptempfiles	Prevents the deletion of temporary files after they are used. If an assembly language file is created by the compiler, this option will place it in the current directory instead of the temporary directory. Produces an object file (called input-file.o) for each source file.
-list	Creates a listing by using the name of the object file with the .lst extension. Assembler option
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DDISABLE_MCAL_INTERMODULE_ASR_CHECK	-D defines a preprocessor symbol to disable the inter-module version check for AR_RELEASE versions. DISABLE_MCAL_INTERMODULE_ASR_CHECK: By default in the package, drivers are compiled to perform the inter-module version check as per Autosar BSW004. When the inter-module version check needs to be disabled then the DISABLE_MCAL_INTERMODULE_ASR_CHECK global define must be added to the list of compiler options.
-DGHS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the GHS preprocessor symbol.

Table 3-2. Assembler Options

Option	Description
-cpu=cortexm4	Selects target processor: Arm Cortex M4
-cpu=cortexm0plus	Selects target processor: Arm Cortex M0+
-c	Produces an object file (called input-file.o) for each source file.
-preprocess_assembly_files	Preprocesses assembly files
-asm=list	Creates a listing by using the name of the object file with the .lst extension. Assembler option

Table 3-3. Linker Options

Option	Description
-Mn	Map file numeric ordering
-delete	Removal from the executable of functions that are unused and unreferenced
-v	Display removed unused functions
-ignore_debug_references	Ignores relocations from DWARF debug sections when using -delete.
-map	Creates a detailed map file
-keepmap	Keep the map file in the event of a link error
-lstartup	Link libstartup library -Run-time environment startup routines
-lsys	Link libsys library -Run-time environment system routines
-larch	Link libarch library -Target-specific run-time support. Any file produced by the Green Hills Compiler may depend on symbols in this library.
-lansi	Link libansi library -the standard C library
-L(/lib/thumb2)	Link thumb2 library
-lutf8_s32	Include utf8_s32.a to use the Wide Character Functions

3.1.2 GCC Compiler/Linker/Assembler Options

Table 3-4. Compiler Options

Option	Description
-c	Produces an object file (called input-file.o) for each source file.
-Os	Use optimization for size.
-ggdb3	Produce debugging information for use by GDB. Level 3 includes extra information, such as all the macro definitions present in the program.
-mcpu=cortex-m4	Selects target processor: Arm Cortex M4
-mcpu=cortex-m0plus	Selects target processor: Arm Cortex M0+
-mthumb	Selects generating code that executes in Thumb state.
-ansi	Specifies ANSI C with extensions.
-mlittle-endian	Generate code for a processor running in little-endian mode.
-fomit-frame-pointer	Removes the frame pointer for all functions, which might make debugging harder.
-msoft-float	Use software floating-point instructions.

Table continues on the next page...

Table 3-4. Compiler Options (continued)

Option	Description
-fno-common	Specifies that the compiler should place uninitialized global variables in the data section of the object file, rather than generating them as common blocks.
-Wall	Enables all the warnings about constructions that some users consider questionable, and that are easy to avoid even in conjunction with macros.
-Wextra	Enables some extra warning flags that are not enabled by '-Wall'.
-Wstrict-prototypes	Warn if a function is declared or defined without specifying the argument types.
-Wno-sign-compare	Do not warn when a comparison between signed and unsigned values could produce an incorrect result when the signed value is converted to unsigned.
-fstack-usage	Geneates an extra file that specifies the maximum amount of stack used, on a per-function basis.
-fdump-ipa-all	Enables all inter-procedural analysis dumps.
-Werror=implicit-function-declaration	Generates an error when the prototype of the function is not defined..
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DGCC	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the GCC preprocessor symbol.

Table 3-5. Assembler Options

Option	Description
-mcpu=cortex-m4	Selects target processor: Arm Cortex M4
-mcpu=cortex-m0plus	Selects target processor: Arm Cortex M0+
-c	Produces an object file (called input-file.o) for each source file.
-mthumb	This option specifies that the assembler should start assembling Thumb instructions.
-x assembler-with-cpp	Indicates that the assembly code contains C directives and the C preprocessor must be run.

Table 3-6. Linker Options

Option	Description
-Map=filename	Print a link map to the file mapfile.
-T scriptfile	Use scriptfile as the linker script. This script replaces ld's default linker script(rather than adding to it), so commandfile must specify everything necessary to describe the output file.
--disable-newlib-supplied-syscalls -specs=nosys.specs	These options support for using newlib on core M0+
-u _printf_float -u _scanf_float	These options support generating profile report.
-nostartfiles	Do not use the standard system startup files when linking
-e _start	Specify that the program entry point is _start
-static	The --static flag tells the linker to link a static, not a dynamically linked
-lc	The -lc flag tells the linker to link this binary against the C library, which is newlib in our case.

Table continues on the next page...

Table 3-6. Linker Options (continued)

Option	Description
-lnosys	The -lnosys flag tells the linker to link this binary against the "nosys" library
$(TOOLCHAIN_DIR)/arm-none-eabi/newlib/lib/thumb/v6-m $(TOOLCHAIN_DIR)/lib/gcc/arm-none-eabi/6.3.1/thumb/v6-m	Library for core M0+
$(TOOLCHAIN_DIR)/arm-none-eabi/newlib/lib/thumb $(TOOLCHAIN_DIR)/arm-none-eabi/newlib/lib)	Library for core M4

3.1.3 IAR Compiler/Linker/Assembler Options

Table 3-7. Compiler Options

Option	Description
--cpu=Cortex-M4	Selects target processor: Arm Cortex M4
--cpu=Cortex-M0+	Selects target processor: Arm Cortex M0+
--cpu_mode=thumb	Selects generating code that executes in Thumb state.
--endian=little	Specifies the endianess of core: little endian.
-Ohz	Sets the optimization level to High, favoring size.
-c	Produces an object file (called input-file.o) for each source file.
--no_clustering	Disables static clustering optimizations.
--no_mem_idioms	Makes the compiler to not optimize code sequences that clear, set, or copy a memory region.
--no_explicit_zero_opt	Places the zero initialized variables in data section instead of bss.
--debug	Makes the compiler include information in the object modules.
--diag_suppress=Pa050	Suppresses diagnostic messages (warnings) about non-standard line endings.
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DIAR	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the IAR preprocessor symbol.
--require_prototypes	Forces the compiler to verify that all functions have proper prototypes.
--no_wrap_diagnostics	Disables line wrapping of diagnostic messages issued by compiler.
--no_system_include	Disables the automatic search for system include files.
-e	Enables language extensions. This option is needed by FLS driver which uses _packed structures.

Table 3-8. Assembler Options

Option	Description
--cpu=Cortex-M4	Selects target processor: Arm Cortex M4
--cpu=Cortex-M0+	Selects target processor: Arm Cortex M0+
--cpu_mode=thumb	Selects generating code that executes in Thumb state.
-g	Use this option to disable the automatic search for system include files.

Table 3-9. Linker Options

Option	Description
--cpu=Cortex-M4	Selects target processor: Arm Cortex M4
--cpu=Cortex-M0+	Selects target processor: Arm Cortex M0+
--map filename	Produces a map file.
--no_library_search	Disables automatic runtime library search.
--entry _start	Treats the symbol _start as a root symbol and as the start of the application.
--enable_stack_usage	Enables stack usage analysis.
--skip_dynamic_initialization	Suppress dynamic initialization during system startup.
--no_wrap_diagnostics	Disables line wrapping of diagnostic messages issued by linker.
--config	Specifies the configuration file to be used by the linker.

3.2 Files required for Compilation

This section describes the include files required to compile, assemble (if assembler code) and link the SPI driver for S32K14X microcontrollers.

To avoid integration of incompatible files, all the include files from other modules shall have the same AR_MAJOR_VERSION and AR_MINOR_VERSION, i.e. only files with the same AUTOSAR major and minor versions can be compiled.

SPI Files

- ..\Spi _ TS_T40D2M10I2R0 \include\Spi.h
- ..\Spi _ TS_T40D2M10I2R0 \include\Spi_IPW.h
- ..\Spi _ TS_T40D2M10I2R0 \include\Spi_IPW_Types.h
- ..\Spi _ TS_T40D2M10I2R0 \include\Spi_LPspi.h
- ..\Spi _ TS_T40D2M10I2R0 \include\Reg_eSys_LPspi.h
- ..\Spi_TS_T40D2M10I2R0\src\Spi.c
- ..\Spi_TS_T40D2M10I2R0\src\Spi_LPspi_Irq.c
- ..\Spi_TS_T40D2M10I2R0\src\Spi_LPspi.c

SPI Generated Files

- Spi_Cfg.c (For PC Variant) - This file should be generated by the user using a configuration tool for compilation.
- Spi_[VariantName]_PBcfg.c (For PB Variant) - This file should be generated by the user using a configuration tool for compilation. The file contains the definition of the init pointer for the respective variant.
- Spi_Cfg.h - This file should be generated by the user using a configuration tool for compilation.

Note

As a deviation from standard:

- Spi_[VariantName]_PBcfg.c - This file will contain the definition for all parameters that are variant aware, independent of the configuration class that will be selected (PC, LT, PB)
- Spi_Cfg.c - This file will contain the definition for all configuration structures containing only variables that are not variant aware, configured and generated only once. This file alone does not contain the whole structure needed by Spi_Init function to configure the driver. Based on the number of variants configured in the EcuC, there can be more than one configuration structure for one module even for PreCompile variant.

Files from Base common folder

- ..\Base_ TS_T40D2M10I2R0 \include\Cer.h
- ..\Base_ TS_T40D2M10I2R0 \include\Compiler.h
- ..\Base_ TS_T40D2M10I2R0 \include\Compiler_Cfg.h
- ..\Base_ TS_T40D2M10I2R0 \include\ComStack_Cfg.h
- ..\Base_ TS_T40D2M10I2R0 \include\ComStack_Types.h
- ..\Base_ TS_T40D2M10I2R0 \include\Mcal.h
- ..\Base_ TS_T40D2M10I2R0 \include\Spi_MemMap.h
- ..\Base_ TS_T40D2M10I2R0 \include\Platform_Types.h
- ..\Base_ TS_T40D2M10I2R0 \include\Reg_eSys.h
- ..\Base_ TS_T40D2M10I2R0 \include\RegLockMacros.h
- ..\Base_ TS_T40D2M10I2R0 \include\SilRegMacros.h
- ..\Base_ TS_T40D2M10I2R0 \include\Soc_Ips.h
- ..\Base_ TS_T40D2M10I2R0 \include\Std_Types.h
- ..\Base_ TS_T40D2M10I2R0 \include\StdRegMacros.h
- ..\Base_ TS_T40D2M10I2R0 \generate_PC\include\StdRegMacros.h

Files from Dem folder:

- ..\Dem_ TS_T40D2M10I2R0 \include\Dem.h
- ..\Dem_ TS_T40D2M10I2R0 \include\Dem_Types.h
- ..\Dem_ TS_T40D2M10I2R0 \generate_PC\include\Dem_IntErrId.h

Files from Det folder:
- ..\Det_ TS_T40D2M10I2R0 \include\Det.h
- ..\Det_ TS_T40D2M10I2R0 \src\Det.c

Files from MCL folder (Only when DMA is used):
- ..\Mcl_ TS_T40D2M10I2R0 \include\CDD_Mcl.h

Files from SchM folder(Only when OS is used):
- ..\Rte_TS_T40D2M10I2R0\include\SchM_Spi.h
- ..\Rte_TS_T40D2M10I2R0\src\SchM_Spi.c

3.3 Setting up the Plug-ins

The SPI driver was designed to be configured by using the EB Tresos Studio (version EB tresos Studio 23.0.0 b170330-0431 or later.)

Location of various files inside the SPI module folder:
- VSMD (Vendor Specific Module Definition) file in EB tresos Studio XDM format:
  - ..\Spi_TS_T40D2M10I2R0\config\Spi.xdm
- Code Generation Templates for Pre-Compile time configuration parameters:
  - ..\Spi_TS_T40D2M10I2R0\generate_PC\src\Spi_Cfg.c
  - ..\Spi_TS_T40D2M10I2R0\generate_PC\include\Spi_Cfg.h
- Code Generation Templates for Post-Build time configuration parameters:
  - ..\Spi_TS_T40D2M10I2R0\generate_PB\src\Spi_[variant]_PBcfg.c

Steps to generate the configuration:
1. Copy the module folders Spi _ TS_T40D2M10I2R0 , Mcu_ TS_T40D2M10I2R0, Mcl_ TS_T40D2M10I2R0 , Base_ TS_T40D2M10I2R0 , Resource_ TS_T40D2M10I2R0 , EcuM_ TS_T40D2M10I2R0 , EcuC_ TS_T40D2M10I2R0 , Rte_ TS_T40D2M10I2R0 , Dem_ TS_T40D2M10I2R0 , Det_ TS_T40D2M10I2R0 into the Tresos plugins folder.
2. Set the desired Tresos Output location folder for the generated sources and header files.
3. Use the EB tresos Studio GUI to modify ECU configuration parameters values.
4. Generate the configuration files.

Dependencies

- **MCU** is required to use System Clock when clock source is used as Peripheral clock source to generate CAN Segment values.
- **RESOURCE** is required to select processor derivative. Current Can driver has support for the following derivatives, everyone having attached a Resource file: s32k148_lqfp144, s32k148_lqfp176, s32k148_mapbga100, s32k146_lqfp144, s32k146_lqfp100, s32k146_lqfp64, s32k146_mapbga100, s32k144_lqfp100, s32k144_lqfp64, s32k144_mapbga100, s32k142_lqfp100, s32k142_lqfp64, s32k118_lqfp48, s32k118_lqfp64, s32k142_lqfp48, s32k144_lqfp48, s32k148_lqfp100 .
- **DET** is required for signalling the development error detection (parameters out of range, null pointers, etc).
- **DEM** is required for signalling the production error detection (hardware failure, etc).
- **MCL** is required when DMA option is used.
- **ECUC** is required for configuring the variant handling in Tresos.

3.3.1 DMA configuration

This section applies only to SPI units configured for asynchronous transmission (SpiPhyUnitSync not checked) and which use DMA for serializing/deserializing data between the hardware unit and the TX/RX buffers (SpiPhyUnitAsyncMethod = DMA).

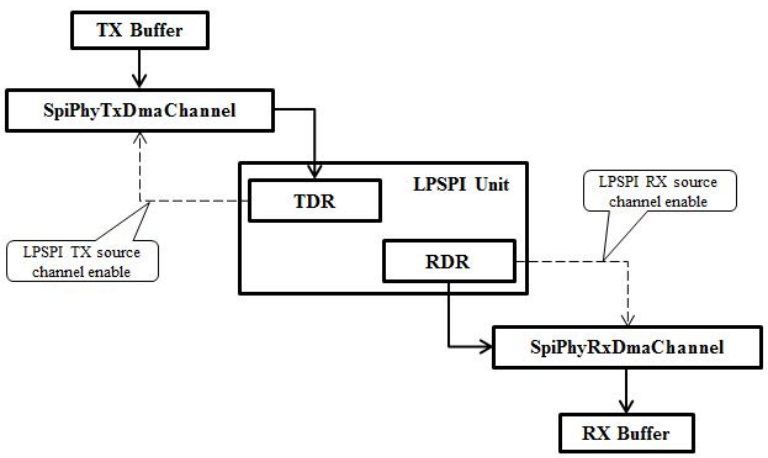

Figure 3-1. DMA transferring mode internal architecture

Each SPI unit configured in DMA mode requires 2 distinct DMA channels from the **same** DMA Mux:

-**SpiPhyTxDmaChannel:** the TX DMA channel used for reading data from the TX buffer and sending to the SPI unit TDR register. This channel is triggered by TX SPI unit event and must be “wired” to “SPI TX source” (configured inside the MCL module – MclDMA folder) – it must be a channel linkable to the external DMA TX source for the given SPI unit.

-**SpiPhyRxDmaChannel:** the RX DMA channel used for filling RX buffer with the deserialized data; this channel is triggered by RX SPI unit event and must be “wired” to “SPI RX source” (configured inside the MCL module – MclDMA folder) – it must be a channel linkable to the external DMA RX source for the given SPI unit.

Note

- If DMA uses fixed priority arbitration, then **SpiPhyRxDmaChannel** priority must be greater than **SpiPhyTxDmaChannel** priority.
- If DMA uses round robin arbitration, no priority constraints are applied on **SpiPhyRxDmaChannel** and **SpiPhyTxDmaChannel** priority.

If the SPI driver is working in interrupt mode, the DMA Rx notification must be enabled for the specified Rx DMA channel and the DMA Tx notification must be enabled for the specified Tx DMA channel. The name of the function to be used as a notification is Spi_LPspi_IsrRxDma_LPSPI_X and Spi_LPspi_IsrTxDma_LPSPI_X, where X is the number of LPSPI unit used.

Next figures show an example of DMA configuration for LPSPI0 unit.

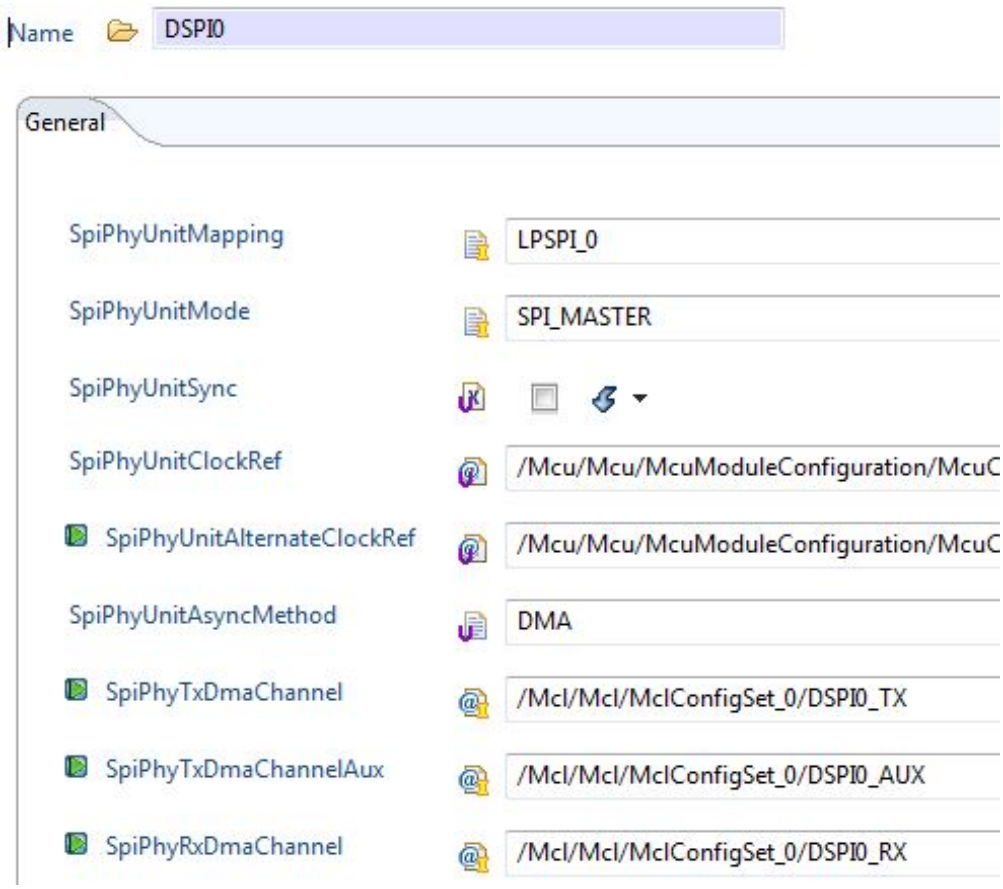

Figure 3-2. DMA Configuration sample for LPSPI0 Physical Unit - SPI module in tresos

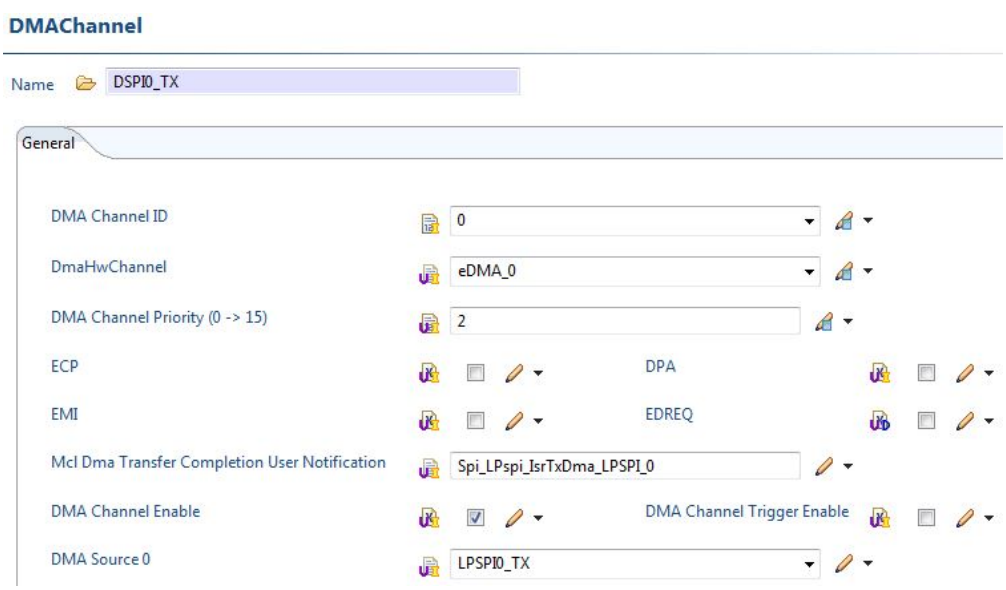

Figure 3-3. DMA channels enable sample for LPSPI0 Physical Unit - MCL module in tresos

Chapter 4
Function calls to module

4.1 Function Calls during Start-up

SPI shall be initialized during STARTUP phase of EcuM initialization. The API to be called for this is Spi _Init(). The MCU module should be initialized before the SPI is initialized. The API to be called for this purpose is Spi_Init(). The PORT and MCL (if the DMA option is used) modules shall be initialized before SPI is initialized.

4.2 Function Calls during Shutdown

SPI can be silenced by calling Spi_DeInit().

4.3 Function Calls during Wake-up

N/A

Chapter 5
Module requirements

5.1 Exclusive areas to be defined in BSW scheduler

SPI_EXCLUSIVE_AREA_01: Used in function Spi_SyncTransmit, to protect the status of the given sequence result. Also it protects the global variable which contains the status of the Spi_SyncTransmit service. As stated by the Autosar, this service cannot be called when another sequence is during transmission, using this service.

SPI_EXCLUSIVE_AREA_02: Used in function Spi_SyncTransmit, to protect the status of the given sequence result. Also it protects the global variable which contains the status of the Spi_SyncTransmit service. As stated by the Autosar, this service cannot be called when another sequence is during transmission, using this service.

SPI_EXCLUSIVE_AREA_03: Used in the internal function Spi_ScheduleJob, protects the schedule mechanism for the situation when a scheduling operation determined by a pending Spi_AsyncTransmit() call may be preempted by a job scheduling requested by an ISR event. It also protect concurrent Spi_AsyncTransmit() calls to schedule in the same time different jobs on the same SPI unit.

SPI_EXCLUSIVE_AREA_04: Used in the internal function Spi_ScheduleNextJob, protects the schedule mechanism for the situation when a scheduling operation determined by a pending Spi_AsyncTransmit() call may be preempted by a job scheduling requested by an ISR event.

SPI_EXCLUSIVE_AREA_05: Used in the internal function Spi_LockJobs, guaranties the atomicity of locking for the entire set of jobs belonging to an asynchronous sequence.

SPI_EXCLUSIVE_AREA_06: Used in the internal function Spi_UnlockRemainingJobs, guaranties the atomicity of unlocking for the entire set of jobs belonging to an asynchronous sequence.

Critical Region Exclusive Matrix

Below is the table depicting the exclusivity between different critical region IDs from the SPI driver. If there is an “X” in a table, it means that those 2 critical regions cannot interrupt each other.

The critical regions from interrupts are grouped in “Interrupt Service Routines Critical Regions (composed diagram)”. If an exclusive area is “exclusive” with the composed “Interrupt Service Routines Critical Regions (composed diagram)” group, it means that it is exclusive with each one of the ISR critical regions.

Table 5-1. Exclusive Areas

	SPI_EXCLUSIVE_AREA_01	SPI_EXCLUSIVE_AREA_02	SPI_EXCLUSIVE_AREA_03	SPI_EXCLUSIVE_AREA_04	SPI_EXCLUSIVE_AREA_05	SPI_EXCLUSIVE_AREA_06	Interrupt Service Routines Critical Regions(composed diagram)
SPI_EXCLUSIVE_AREA_01	X	X			X	X	X
SPI_EXCLUSIVE_AREA_02	X	X			X	X	X
SPI_EXCLUSIVE_AREA_03			X	X	X	X	X
SPI_EXCLUSIVE_AREA_04			X	X	X	X	X
SPI_EXCLUSIVE_AREA_05	X	X	X	X	X	X	X
SPI_EXCLUSIVE_AREA_06	X	X	X	X	X	X	X
Interrupt Service Routines Critical Regions (composed diagram)	X	X	X	X	X	X	X

5.2 Peripheral Hardware Requirements

N/A

5.3 ISR to configure within OS – dependencies

The following ISRs are used by the SPI driver and need to be assigned to a priority level. The interrupt vector numbers corresponding to PIO_FIFO for master&slave mode is as shown in bottom table. Int the master mode, interrupt occurs each time the TDF or RDF bits in SR register arises, and occurs each time the RDF bit in SR register arises with slave mode. The interrupt vector numbers of DMA channel configuration depend on the number of used DMA channel in EB Tresos configuration for SPI modules. Please see details in the reference manual.

Note

- Unused interrupts shouldn't be configured in the OS.

Table 5-2. SPI ISRs for DMA

Physical Unit	ISR Name
LPSPI_0	Spi_LPspi_IsrRxDma_LPSPI_0
LPSPI_0	Spi_LPspi_IsrTxDma_LPSPI_0
LPSPI_1	Spi_LPspi_IsrRxDma_LPSPI_1
LPSPI_1	Spi_LPspi_IsrTxDma_LPSPI_1
LPSPI_2	Spi_LPspi_IsrRxDma_LPSPI_2
LPSPI_2	Spi_LPspi_IsrTxDma_LPSPI_2

Table 5-3. SPI ISRs for PIO_FIFO in Master mode

Physical Unit	ISR Name	Hardware interrupt vector
LPSPI_0	Spi_LPspi_IsrTDF_LPSPI_0	26
LPSPI_1	Spi_LPspi_IsrTDF_LPSPI_1	27
LPSPI_2	Spi_LPspi_IsrTDF_LPSPI_2	28

Table 5-4. SPI ISRs for PIO_FIFO in Slave mode

Physical Unit	ISR Name	Hardware interrupt vector
LPSPI_0	Spi_LPspi_IsrTDF_LPSPI_0	26
LPSPI_1	Spi_LPspi_IsrTDF_LPSPI_1	27
LPSPI_2	Spi_LPspi_IsrTDF_LPSPI_2	28

Note: In case of AUTOSAR_OS_NOT_USED, the compiler option "-DUSE_SW_VECTOR_MODE" must be added to the list of compiler options to be used with interrupt controller configured to be in software vector mode.

5.4 ISR Macro

MCAL drivers use the ISR macro to define the functions that will process hardware interrupts. Depending on whether the OS is used or not, this macro can have different definitions:

a. OS is not used - AUTOSAR_OS_NOT_USED is defined:

i. If USE_SW_VECTOR_MODE is defined:

```
#define ISR(IsrName) void IsrName(void)
```

In this case, drivers' interrupt handlers are normal C functions and the prolog/epilog handle the context save and restore.

ii. If USE_SW_VECTOR_MODE is not defined:

```
#define ISR(IsrName) INTERRUPT_FUNC void IsrName(void)
```

In this case, drivers' interrupt handlers must save and restore the execution context.

Custom OS is used - AUTOSAR_OS_NOT_USED is not defined

```
#define ISR(IsrName) void OS_isr_##IsrName()
```

In this case, OS is handling the execution context when an interrupt occurs. Drivers' interrupt handlers are normal C functions.

Other vendor's OS is used - AUTOSAR_OS_NOT_USED is not defined. Please refer to the OS documentation for description of the ISR macro.

5.5 Other AUTOSAR modules - dependencies

BASE:

The BASE module contains the common files/definitions needed by all MCAL modules.

DEM :

The DEM module is used for enabling reporting of production relevant error status. The API function used is Dem_ReportErrorStatus().

Resource :

Resource module is used to select microcontroller's derivatives.

RTE :

The RTE module is needed for implementing data consistency of exclusive areas that are used by SPI module.

DET:

The DET module is used for enabling Default Error Tracer detection. The API function used is Det_ReportError(). The activation / deactivation of Default Error Tracer detection is configurable using the

'SpiDevErrorDetect' configuration parameter.

ECUC:

The ECUC module is used for ECU configuration. MCAL modules need ECUC to retrieve the variant information.

PORT :

The PORT module is used to configure the port pins with the needed modes, before they are used by the SPI module. For each SPI, the SCK, SOUT, SIN and CSx_y signals need to be configured. In the S32K14X Reference manual there is an example of the pin configuration. Please refer to the Reference List.

MCU:

The MCU driver provides services for basic microcontroller initialization, power down functionality, reset and microcontroller specific functions required by other MCAL software modules. The clocks need to be initialized prior to using the SPI driver. The SPI reference clock is provided by MCU plugin. The clock frequency may affect the Baudrate, Timing between clock and chip select, Timing between chip select and clock, Timing between chip select assertions. The reference is specified by the parameter SpiGeneral\SpiPhyUnit\SpiPhyUnitClockRef:

Figure 5-1. Spi reference clock provided by MCU plugin

MCL:

For each LPSPI in use, a transmit and a receive DMA channel need to be defined and routed through the DMA Multiplexer using MCL plugin. MCL should be initialized before SPI switch to DMA mode

The Table Table 5-5 shows an example DMA configuration. For more information, refer to section DMA configuration

Table 5-5. SPI DMA Channel Multiplexer

DMA Name	DMA Source 0
LPSPI 0 Transmit DMA	15 (LPSPI0.SpiPhyTxDmaChannel)
LPSPI 0 Receive DMA	14 (LPSPI0.SpiPhyRxDmaChannel)
LPSPI 1 Transmit DMA	17 (LPSPI1.SpiPhyTxDmaChannel)
LPSPI 1 Receive DMA	16 (LPSPI1.SpiPhyRxDmaChannel)
LPSPI 2 Transmit DMA	19 (LPSPI2.SpiPhyTxDmaChannel)
LPSPI 2 Receive DMA	18 (LPSPI2.SpiPhyRxDmaChannel)

5.6 Data Cache Restriction

In the DMA transfer mode, DMA transfers may issue cache coherency problems. To avoid possible coherency issues when **D-CACHE** is enabled, the user shall ensure that the buffers used as TCD source and destination are allocated in the **NON-CACHEABLE** area (by means of Memmap). Otherwise, the SPI driver has some dependencies. The user must follow the below things:

The first: Should not use the internal buffer for transmitter and receiver

The second: User must to put all variables, which were used for transmitter and receiver, to the **NON CACHEABLE** memory section in the RAM zone by the definition **SPI_START_SEC_VAR_<INIT_POLICY>_<ALIGNMENT>_NO_CACHEABLE** and **SPI_STOP_SEC_VAR_<INIT_POLICY>_<ALIGNMENT>_NO_CACHEABLE**

5.7 User Mode Support

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

Chapter 6
Main API Requirements

6.1 Main functions calls within BSW scheduler

The function Spi_MainFunction_Handling() should be called periodically only if polling mode is enabled for Spi_AsyncTransmit() .

6.2 API Requirements

None

6.3 Calls to Notification Functions, Callbacks, Callouts

Call-back Notifications:

None.

User Notification:

The SPI Handler & Driver provides notifications per job and sequence in asynchronous mode. The notifications can be configured as pointers to user defined functions. If notification is not desired, the appropriate EndNotification field shall be left blank.

For asynchronous transmissions, job and sequences notifications are performed before the scheduling of the next job (contrary to the recommendation given by SPI088) . In this way, calls like Spi_SetupIB() or Spi_WriteIB() can be targeted on the next schedulable jobs, before the starting of the job transfer.

Chapter 7
Memory Allocation

7.1 Sections to be defined in Spi_MemMap.h

Table 7-1. Memory Allocation

Section name	Type of section	Description
SPI_START_SEC_CONFIG_DATA_UNSPECIFIED	Configuration Data	Start of Memory Section for Config Data
SPI_STOP_SEC_CONFIG_DATA_UNSPECIFIED	Configuration Data	End of Memory Section for Config Data
SPI_START_SEC_CONST_32	Configuration Data	Start of Memory Section for Config Data.
SPI_STOP_SEC_CONST_32	Configuration Data	End of Memory Section for Config Data
SPI_START_SEC_CODE	Code	Start of memory Section for Code
SPI_STOP_SEC_CODE	Code	End of memory Section for Code
SPI_START_SEC_VAR_NO_INIT_32	Variables	Used for variables which have to be aligned to 32 bit. For instance used for variables of size 32 bit or used for composite data types: arrays, structs containing elements of maximum 32 bits. These variables are never cleared and never initialized by start-up code.
SPI_STOP_SEC_VAR_NO_INIT_32	Variables	End of above section.
SPI_START_SEC_VAR_NO_INIT_UNSPECIFIED	Variables	Used for variables, structures, arrays when the SIZE (alignment) does not fit the criteria of 8,16 or 32 bit. These variables are never cleared and never initialized by start-up code.
SPI_STOP_SEC_VAR_NO_INIT_UNSPECIFIED	Variables	End of above section.
SPI_START_SEC_VAR_NO_INIT_UNSPECIFIED_NO_CACHEABLE	Variables	Used for variables, structures, arrays when the SIZE (alignment) does not fit the criteria of 8,16 or 32 bit, and that have to be stored in a non-cacheable memory section. These variables are never cleared and never initialized by start-up code.
SPI_STOP_SEC_VAR_NO_INIT_UNSPECIFIED_NO_CACHEABLE	Variables	End of above section.

Table continues on the next page...

Table 7-1. Memory Allocation (continued)

Section name	Type of section	Description
SPI_START_SEC_VAR_INIT_32	Variables	Used for variables which have to be aligned to 32 bit. For instance used for variables of size 32 bit or used for composite data types: arrays ,structs containing elements of maximum 32 bits. These variables are initialized with values after every reset.
SPI_STOP_SEC_VAR_INIT_32	Variables	End of above section.
SPI_START_SEC_VAR_INIT_UNSPECIFIED	Variables	Used for variables, structures, arrays, when the SIZE (alignment) does not fit the criteria of 8,16 or 32 bit. These variables are initialized with values after every reset.
SPI_STOP_SEC_VAR_INIT_UNSPECIFIED	Variables	End of above section.
SPI_START_SEC_CONST_32	Constant Data	Used for constants that have to be aligned to 32 bit.
SPI_STOP_SEC_CONST_32	Constant Data	End of above section.

7.2 Linker command file

Memory shall be allocated for every section defined in SPI_MemMap.h

Chapter 8
Configuration parameters considerations

Configuration parameter class for Autosar SPI driver fall into the following variants as defined below:

8.1 Configuration Parameters

Configuration parameter class for Autosar SPI driver fall into the following variants as defined below:

Table 8-1. Configuration Parameters

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
SpiDriver	SPI_MAX_CHANNEL	PC, LT or PB	Pre Compile (1)
	SPI_MAX_JOB	PC, LT or PB	Pre Compile (1)
	SPI_MAX_SEQUENCE	PC, LT or PB	Pre Compile (1)
SpiChannel	SpiChannelId	Pre-Compile all Variants	Pre Compile
	SpiChannelType	PC, LT or PB	Pre Compile (2)
	SpilbNBuffers	PC, LT or PB	Pre Compile (3)
	SpiDataWidth	PC, LT or PB	Post Build
	SpiDefaultData	PC, LT or PB	Post Build
	SpiEbMaxlength	PC, LT or PB	Post Build
	SpiTransferStart	PC, LT or PB	Post Build
SpiDemEventParameterRefs	Spi_E_Hardware_Error	PC, LT or PB	Post Build
SpiExternalDevice	SpiSlaveMode	PC, LT or PB	Post Build
	TSBModeEnable	PC, LT or PB	Post Build
	ITSBModeEnable	PC, LT or PB	Post Build
	SpiBaudRate	PC, LT or PB	Post Build
	SpiEnableCs	PC, LT or PB	Post Build
	SpiCsIdentifier	PC, LT or PB	Post Build
	SpiCsPolarity	PC, LT or PB	Post Build
	SpiCsSelection	PC, LT or PB	Post Build
	SpiDataShiftEdge	PC, LT or PB	Post Build

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

	SpiHwUnit	PC, LT or PB	Post Build
	SpiShiftClockIdleLevel	PC, LT or PB	Post Build
	SpiTimeClk2Cs	PC, LT or PB	Post Build
	SpiTImeCs2Clk	Vendor specific	Post Build
	SpiTimeCs2Cs	Vendor specific	Post Build
	SpiCsContinuous	Vendor specific	Post Build
SpiJob	TSBModeEnable	Pre-Compile all Variants	Pre Compile
	ITSBModeEnable	Pre-Compile all Variants	Pre Compile
	SpiHwUnitSynchronous	PC, LT or PB	Post Build
	SpiJobEndNotification	PC, LT or PB	Post Build
	SpiJobStartNotification	PC, LT or PB	Post Build
	SpiJobId	Pre-Compile all Variants	Pre Compile
	SpiJobPriority	PC, LT or PB	Post Build
	SpiDeviceAssignment	PC, LT or PB	Post Build
	TSBFrameSize	PC, LT or PB	Post Build
	TS0_LEN	PC, LT or PB	Post Build
	TS1_LEN	PC, LT or PB	Post Build
	TS2_LEN	PC, LT or PB	Post Build
	TS3_LEN	PC, LT or PB	Post Build
	TS0_CONF	PC, LT or PB	Post Build
	TS1_CONF	PC, LT or PB	Post Build
	TS2_CONF	PC, LT or PB	Post Build
	TS3_CONF	PC, LT or PB	Post Build
	DsiCsIdentifier	PC, LT or PB	Post Build
	TransmitDataSource	PC, LT or PB	Post Build
	ChangeInDataTransfer	PC, LT or PB	Post Build
	DualReceiverSupport	Pre-Compile all Variants	Pre Compile
	SecondaryFrameSize	PC, LT or PB	Post Build
	SpiChannelAssignment	PC, LT or PB	Post Build
	SecondaryDsiCsIdentifier	PC, LT or PB	Post Build
	SpiSequenceId	Pre-Compile all Variants	Pre Compile
SpiSequence	SpiInterruptibleSequence	PC, LT or PB	Post Build
	SpiSeqEndNotification	PC, LT or PB	Post Build
	SpiJobAssignment	PC, LT or PB	Post Build
SpiGeneral	SpiCancelApi	Pre-Compile all Variants	Pre Compile
	SpiChannelBuffersAllowed	Pre-Compile all Variants	Pre Compile
	SpiDevErrorDetect	Pre-Compile all Variants	Pre Compile
	SpiHwStatusApi	Pre-Compile all Variants	Pre Compile
	SpiInterruptibleSeqAllowed	Pre-Compile all Variants	Pre Compile
	SpiLevelDelivered	Pre-Compile all Variants	Pre Compile

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

	SpiSupportConcurrentSyncTransmit	Vendor specific	Pre Compile
	SpiVersionInfoApi	Pre-Compile all Variants	Pre Compile
	SpiClockRef	Vendor specific	Pre Compile (4)
	SpiGlobalDmaEnable	Vendor specific	Pre Compile
	SpiSyncTransmitTimeout	Vendor specific	Pre Compile
	SpiOptimizeOneJobSequences	Vendor specific	Pre Compile
	SpiOptimizedSeqNumber	Vendor specific	Pre Compile
	SpiOptimizedChannelsNumber	Vendor specific	Pre Compile
SpiNonAUTOSAR	SpiAllowBigSizeCollections	Vendor specific	Pre Compile
	SpiEnableHWUnitAsyncMode	Vendor specific	Pre Compile
	SpiTSBModeSupport	Vendor Specific	Pre Compile
	SpiITSBModeSupport	Vendor Specific	Pre Compile
	SpiEnableDualClockMode	Vendor specific	Pre Compile
	SpiJobStartNotificationenable	Vendor specific	Pre Compile
	SpiForceDataType	Vendor specific	Pre Compile
	SpiDisableDemReportErrorStatus	Vendor specific	Pre Compile
SpiPhyUnit	SpiPhyUnitMapping	Vendor specific	Pre Compile
	SpiPhyUnitMode	Vendor specific	Post Build
	SpiPhyUnitSync	Vendor specific	Post Build
	SpiPhyUnitClockRef	Vendor specific	Post Build
	SpiPhyUnitAlternateClockRef	Vendor specific	Post Build
	SpiPhyUnitAsyncMethod	Vendor specific	Post Build
	SpiPhyTxDmaChannel	Vendor specific	Post Build
	SpiPhyTxDmaChannelAux	Vendor specific	Post Build
	SpiPhyRxDmaChannel	Vendor specific	Post Build

Chapter 9
Integration Steps

This section gives a brief overview of the steps needed for integrating Serial Peripheral Interface :

- Generate the required SPI configurations. For more details refer to section Files required for Compilation
- Allocate proper memory sections in SPI_MemMap.h and linker command file. For more details refer to section
- Compile & build the SPI with all the dependent modules. For more details refer to section Building the Driver

Chapter 10
ISR Reference

ISR functions exported by the SPI driver.

Chapter 11
External Assumptions for SPI driver

The section presents requirements that must be complied with when integrating SPI driver into the application.

[SMCAL_CPR_EXT163]

<< If interrupts are locked a centralized function pair to lock and unlock interrupts shall be used. >>

[SWS_Spi_00239]

<< SPI peripherals may depend on the system clock, prescaler(s) and PLL. Thus, changes of the system clock (e.g. PLL on (PLL off) may also affect the clock settings of the SPI hardware. >>

[SWS_Spi_00244]

<< The SPI Handler/Driver module does not take care of setting the registers which configure the clock, prescaler(s) and PLL in its init function. This has to be done by the MCU module. >>

[SWS_Spi_00052]

<< For the IB Channels, the Handler/Driver shall provide the buffering but it is not able to take care of the consistency of the data in the buffer during transmission. The size of the Channel buffer is fixed. >>

[SWS_Spi_00257]

<< The SPI Handler/Driver is not able to prevent the overwriting of these "transmit" buffers by users during transmissions. >>

[SWS_Spi_00053]

<< For EB Channels the application shall provide the buffering and shall take care of the consistency of the data in the buffer during transmission. >>

[SWS_Spi_00280]

<< The buffer provided by the application for the SPI Handler Driver may have a different size. >>

NOTE

This referes in the context of External Buffer

[SWS_Spi_00084]

<< If different Jobs (and consequently also Sequences) have common Channels, the SPI Handler/Driver' environment shall ensure that read and/or write functions are not called during transmission. >>

[SWS_Spi_00121]

<< The SPI Handler/Driver's environment shall configure the SpiInterruptibleSeqAllowed parameter (ON / OFF) in order to select which kind of Sequences the SPI Handler/Driver manages. >>

[SWS_Spi_00080]

<< When using Interruptible Sequences, the caller must be aware that if the multiple Sequences access the same Channels, the data for these Channels may be overwritten by the highest priority Job accessing each Channel. >>

[SWS_Spi_00298]

<< The operation Spi_Init is Non Re-entrant. >>

[SWS_Spi_00300]

<< The operation Std_ReturnType Spi_DeInit( ) is Non Re-entrant. >>

[SWS_Spi_00173]

<< The SPI Handler/Driver's environment shall call the function Spi_AsyncTransmit after a function call of Spi_SetupEB for EB Channels or a function call of Spi_WriteIB for IB Channels but before the function call Spi_ReadIB. >>

[SWS_Spi_00027]

<< The SPI Handler/Driver's environment shall call the function Spi_ReadIB after a Transmit method call to have relevant data within IB Channel. >>

[SWS_Spi_00037]

<< The SPI Handler/Driver's environment shall call the Spi_SetupEB function once for each Channel with EB declared before the SPI Handler/Driver's environment calls a Transmit method on them. >>

[SWS_Spi_00038]

<< The SPI Handler/Driver's environment shall call the function Spi_GetJobResult to inquire whether the Job transmission has succeeded (SPI_JOB_OK) or failed (SPI_JOB_FAILED). >>

[SWS_Spi_00042]

<< The SPI Handler/Driver's environment shall call the function Spi_GetSequenceResult to inquire whether the full Sequence transmission has succeeded (SPI_SEQ_OK) or failed (SPI_SEQ_FAILED). >>

[SWS_Spi_00325]

<< The operation Spi_GetVersionInfo is Non Re-entrant. >>

[SWS_Spi_00287]

<< The SPI Handler/Driver's environment shall call this function to inquire whether the specified SPI Hardware microcontroller peripheral is SPI_IDLE or SPI_BUSY. >>

NOTE

This requirement refers to Spi_GetHWUnitStatus()

[SWS_Spi_00335]

<< The operation Spi_SetAsyncMode is Non Re-entrant. >>

[SWS_Spi_00265]

<< For implement the call back function other modules are required to provide the routines in the expected manner. >>

[SWS_Spi_00048]

<< The callback notifications Spi_JobEndNotification and Spi_SeqEndNotification shall have no parameters and no return value. >>

[SWS_Spi_00085]

<< It is allowed to use the following API calls within the SPI callback notifications:

Spi_ReadIB

Spi_WriteIB

Spi_SetupEB

Spi_GetJobResult

Spi_GetSequenceResult

Spi_GetHWUnitStatus

Spi_Cancel

All other SPI Handler/Driver API calls are not allowed. >>

[SWS_Spi_00340]

<< The operation SpiJobEndNotification is Re-entrant. >>

[SWS_Spi_00341]

<< The operation SpiSeqEndNotification is Re-entrant. >>

[SWS_Spi_00077]

<< To transmit a variable number of data, it is mandatory to call the Spi_SetupEB function to store new parameters within SPI Handler/Driver before each Spi_AsyncTransmit function call. >>

[SWS_Spi_00078]

<< To transmit a constant number of data, it is only mandatory to call the Spi_SetupEB function to store parameters within SPI Handler/Driver before the first Spi_AsyncTransmit function call. >>

[SWS_Spi_00235]

<< If not applicable, the SPI Handler/Driver module's environment shall pass a NULL pointer to the function Spi_Init. >>

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.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

Document Number IM2SPIASR4.2 Rev0002R1.0.2
Revision 1.0