

Integration Manual

for S32 PORT Driver

Document Number: IM11PORTASR4.4 Rev0000R4.0.0 Rev. 1.0

1 Revision History	2
2 Introduction	3
2.1 Supported Derivatives	3
2.2 Overview	4
2.3 About This Manual	4
2.4 Acronyms and Definitions	5
2.5 Reference List	5
3 Building the driver	6
3.1 Build Options	6
3.1.1 GCC Compiler/Assembler/Linker Options	6
3.1.2 GHS Compiler/Assembler/Linker Options	9
3.1.3 DIAB Compiler/Assembler/Linker Options	11
3.2 Files required for compilation	13
3.3 Setting up the plugins	15
4 Function calls to module	16
4.1 Function Calls during Start-up	16
4.2 Function Calls during Shutdown	16
4.3 Function Calls during Wake-up	16
5 Module requirements	17
5.1 Exclusive areas to be defined in BSW scheduler	17
5.2 Exclusive areas not available on this platform	18
5.3 Peripheral Hardware Requirements	18
5.4 ISR to configure within AutosarOS - dependencies	18
5.5 ISR Macro	18
5.5.1 Without an Operating System	18
5.5.2 With an Operating System	19
5.6 Other AUTOSAR modules - dependencies	19
5.7 Data Cache Restrictions	19
5.8 User Mode support	20
5.8.1 User Mode configuration in the module	20
5.8.2 User Mode configuration in AutosarOS	20
5.9 Multicore support	21
6 Main API Requirements	23
6.1 Main function calls within BSW scheduler	23
6.2 API Requirements	23
6.3 Calls to Notification Functions, Callbacks, Callouts	23

7 Memory allocation	24
7.1 Sections to be defined in Port_MemMap.h	24
7.2 Linker command file	25
8 Integration Steps	26
9 External assumptions for driver	27



Chapter 1

Revision History

Revision	Date	Author	Description
1.0	31.10.2022	NXP RTD Team	Prepared for release S32 RTD AUTOSAR 4.4 Version 4.0.0 Release

Chapter 2

Introduction

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

This integration manual describes the integration requirements for Port Driver for S32 microcontrollers.

2.1 Supported Derivatives

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

- s32g274a_bga525
- s32g254a_bga525
- s32g233a_bga525
- s32g234m_bga525
- s32g378a_bga525
- s32g379a_bga525
- s32g398a_bga525
- s32g399a_bga525
- s32g338m_bga525
- s32g339m_bga525
- s32g358a_bga525
- s32g359a_bga525
- s32r45_bga780

All of the above microcontroller devices are collectively named as S32.

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
ASM	Assembler
BSMI	Basic Software Make file Interface
CAN	Controller Area Network
C/CPP	C and C++ Source Code
CS	Chip Select
CTU	Cross Trigger Unit
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
VLE	Variable Length Encoding
XML	Extensible Markup Language

2.5 Reference List

#	Title	Version
1	Specification of Port Driver	AUTOSAR Release 4.4.0
2	Reference Manual	S32G2 Reference Manual, Rev 5, May 2022
		S32G3 Reference Manual, Rev.2 Draft C, June 2022
		S32R45 Reference Manual, Rev. 3, 12/2021
3	Datasheet	S32G2 Data Sheet, Rev 5, May 2022
		S32G3 Data Sheet, Rev 2, Draft B, June 2022
		VR5510 Data Sheet, Rev 5, April 2022
		S32R45 Data Sheet, Rev. 2 — 12/2021
4	Errata	S32G2: Mask Set Errata for Mask 0P77B, Rev. 2.4
		S32G3: Mask Set Errata for Mask 0P72B, Rev. 1.1
		S32R45: Mask Set Errata for Mask P57D, Rev. 2.0

Chapter 3

Building the driver

- [Build Options](#)
- [Files required for compilation](#)
- [Setting up the plugins](#)

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

3.1 Build Options

- [GCC Compiler/Assembler/Linker Options](#)
- [GHS Compiler/Assembler/Linker Options](#)
- [DIAB Compiler/Assembler/Linker Options](#)

The RTD driver files are compiled using:

- NXP GCC 9.2.0 20190812 (Build 1649 Revision gaf57174)
- Green Hills Multi 7.1.6d / Compiler 2020.1.4
- Wind River Diab Compiler 7.0.3

The compiler, assembler, and linker flags used for building the driver are explained below.

The TS_T40D11M40I0R0 part of the plugin name is composed as follows:

- T = Target_Id (e.g. T40 identifies Cortex-M architecture)
- D = Derivative_Id (e.g. D11 identifies S32 platform)
- M = SW_Version_Major and SW_Version_Minor
- I = SW_Version_Patch
- R = Reserved

3.1.1 GCC Compiler/Assembler/Linker Options

3.1.1.1 GCC Compiler Options

Compiler Option	Description
-mcpu=cortex-m7	Targeted ARM processor for which GCC should tune the performance of the code
-mthumb	Generates code that executes in Thumb state
-mlittle-endian	Generate code for a processor running in little-endian mode
-mfpv=fpv5-sp-d16	Specifies the floating-point hardware available on the target
-mfloat-abi=hard	Specifies the floating-point ABI to use. "hard" allows generation of floating-point instructions and uses FPU-specific calling conventions
-std=c99	Specifies the ISO C99 base standard
-Os	Optimize for size. Enables all -O2 optimizations except those that often increase code size
-ggdb3	Produce debugging information for use by GDB using the most expressive format available, including GDB extensions if at all possible. Level 3 includes extra information, such as all the macro definitions present in the program
-Wall	Enables all the warnings about constructions that some users consider questionable, and that are easy to avoid (or modify to prevent the warning), even in conjunction with macros
-Wextra	This enables some extra warning flags that are not enabled by -Wall
-pedantic	Issue all the warnings demanded by strict ISO C. Reject all programs that use forbidden extensions. Follows the version of the ISO C standard specified by the aforementioned -std option
-Wstrict-prototypes	Warn if a function is declared or defined without specifying the argument types
-Wundef	Warn if an undefined identifier is evaluated in an #if directive. Such identifiers are replaced with zero
-Wunused	Warn whenever a function, variable, label, value, macro is unused
-Werror=implicit-function-declaration	Make the specified warning into an error. This option throws an error when a function is used before being declared
-Wsign-compare	Warn when a comparison between signed and unsigned values could produce an incorrect result when the signed value is converted to unsigned.
-Wdouble-promotion	Give a warning when a value of type float is implicitly promoted to double
-fno-short-enums	Specifies that the size of an enumeration type is at least 32 bits regardless of the size of the enumerator values.
-funsigned-char	Let the type char be unsigned by default, when the declaration does not use either signed or unsigned
-funsigned-bitfields	Let a bit-field be unsigned by default, when the declaration does not use either signed or unsigned
-fomit-frame-pointer	Omit the frame pointer in functions that don't need one. This avoids the instructions to save, set up and restore the frame pointer; on many targets it also makes an extra register available.

Compiler Option	Description
-fno-common	Makes the compiler place uninitialized global variables in the BSS section of the object file. This inhibits the merging of tentative definitions by the linker so you get a multiple-definition error if the same variable is accidentally defined in more than one compilation unit
-fstack-usage	This option is only used to build test for generation Ram/↔ Stack size report. Makes the compiler output stack usage information for the program, on a per-function basis
-fdump-ipa-all	This option is only used to build test for generation Ram/↔ Stack size report. Enables all inter-procedural analysis dumps
-c	Stop after assembly and produce an object file for each source file
-DS32XX	Predefine S32XX as a macro, with definition 1
-DS32G2XX	Predefine S32G2XX as a macro, with definition 1
-DGCC	Predefine GCC as a macro, with definition 1
-DUSE_SW_VECTOR_MODE	Predefine USE_SW_VECTOR_MODE as a macro, with definition 1. By default, the drivers are compiled to handle interrupts in Software Vector Mode
-DD_CACHE_ENABLE	Predefine D_CACHE_ENABLE as a macro, with definition 1. Enables data cache initialization in source file system.↔ c under the Platform driver
-DI_CACHE_ENABLE	Predefine I_CACHE_ENABLE as a macro, with definition 1. Enables instruction cache initialization in source file system.c under the Platform driver
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initialization in source file system.c under the Platform driver
-DMCAL_ENABLE_USER_MODE_SUPPORT	Predefine MCAL_ENABLE_USER_MODE_SUPPORT as a macro, with definition 1. Allows drivers to be configured in user mode.

3.1.1.2 GCC Assembler Options

Assembler Option	Description
-X assembler-with-cpp	Specifies the language for the following input files (rather than letting the compiler choose a default based on the file name suffix)
-mcpu=cortexm7	Targeted ARM processor for which GCC should tune the performance of the code
-mfpu=fpv5-sp-d16	Specifies the floating-point hardware available on the target
-mfloat-abi=hard	Specifies the floating-point ABI to use. "hard" allows generation of floating-point instructions and uses FPU-specific calling conventions
-mthumb	Generates code that executes in Thumb state
-c	Stop after assembly and produce an object file for each source file

3.1.1.3 GCC Linker Options

Linker Option	Description
-Wl,-Map,filename	Produces a map file
-T linkerfile	Use linkerfile as the linker script. This script replaces the default linker script (rather than adding to it)
-entry=Reset_Handler	Specifies that the program entry point is Reset_Handler
-nostartfiles	Do not use the standard system startup files when linking
-mcpu=cortexm7	Targeted ARM processor for which GCC should tune the performance of the code
-mthumb	Generates code that executes in Thumb state
-mfpu=fpv5-sp-d16	Specifies the floating-point hardware available on the target
-mfloat-abi=hard	Specifies the floating-point ABI to use. "hard" allows generation of floating-point instructions and uses FPU-specific calling conventions
-mlittle-endian	Generate code for a processor running in little-endian mode
-ggdb3	Produce debugging information for use by GDB using the most expressive format available, including GDB extensions if at all possible. Level 3 includes extra information, such as all the macro definitions present in the program
-lc	Link with the C library
-lm	Link with the Math library
-lgcc	Link with the GCC library

3.1.2 GHS Compiler/Assembler/Linker Options

3.1.2.1 GHS Compiler Options

Compiler Option	Description
-cpu=cortexm7	Selects target processor: Arm Cortex M7
-thumb	Selects generating code that executes in Thumb state
-fpu=vfpv5_d16	Specifies hardware floating-point using the v5 version of the VFP instruction set, with 16 double-precision floating-point registers
-fsingle	Use hardware single-precision, software double-precision FP instructions
-C99	Use (strict ISO) C99 standard (without extensions)
-ghstd=last	Use the most recent version of Green Hills Standard mode (which enables warnings and errors that enforce a stricter coding standard than regular C and C++)
-Osize	Optimize for size
-gnu_asm	Enables GNU extended asm syntax support
-dual_debug	Generate DWARF 2.0 debug information
-G	Generate debug information
-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
-Wimplicit-int	Produce warnings if functions are assumed to return int
-Wshadow	Produce warnings if variables are shadowed

Compiler Option	Description
-Wtrigraphs	Produce warnings if trigraphs are detected
-Wundef	Produce a warning if undefined identifiers are used in #if preprocessor statements
-unsigned_chars	Let the type char be unsigned, like unsigned char
-unsigned_fields	Bitfields declared with an integer type are unsigned
-no_commons	Allocates uninitialized global variables to a section and initializes them to zero at program startup
-no_exceptions	Disables C++ support for exception handling
-no_slash_comment	C++ style // comments are not accepted and generate errors
-prototype_errors	Controls the treatment of functions referenced or called when no prototype has been provided
-incorrect_pragma_warnings	Controls the treatment of valid #pragma directives that use the wrong syntax
-c	Stop after assembly and produce an object file for each source file
-DS32XX	Predefine S32XX as a macro, with definition 1
-DS32G2XX	Predefine S32G2XX as a macro, with definition 1
-DGHS	Predefine GHS as a macro, with definition 1
-DUSE_SW_VECTOR_MODE	Predefine USE_SW_VECTOR_MODE as a macro, with definition 1. By default, the drivers are compiled to handle interrupts in Software Vector Mode
-DD_CACHE_ENABLE	Predefine D_CACHE_ENABLE as a macro, with definition 1. Enables data cache initialization in source file system.c under the Platform driver
-DI_CACHE_ENABLE	Predefine I_CACHE_ENABLE as a macro, with definition 1. Enables instruction cache initialization in source file system.c under the Platform driver
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initialization in source file system.c under the Platform driver
-DMCAL_ENABLE_USER_MODE_SUPPORT	Predefine MCAL_ENABLE_USER_MODE_SUPPORT as a macro, with definition 1. Allows drivers to be configured in user mode

3.1.2.2 GHS Assembler Options

Assembler Option	Description
-cpu=cortexm7	Selects target processor: Arm Cortex M7
-fpu=vfpv5_d16	Specifies hardware floating-point using the v5 version of the VFP instruction set, with 16 double-precision floating-point registers
-fsingle	Use hardware single-precision, software double-precision FP instructions
-preprocess_assembly_files	Controls whether assembly files with standard extensions such as .s and .asm are preprocessed
-list	Creates a listing by using the name and directory of the object file with the .lst extension
-c	Stop after assembly and produce an object file for each source file

3.1.2.3 GHS Linker Options

Linker Option	Description
-e Reset_Handler	Make the symbol Reset_Handler be treated as a root symbol and the start label of the application
-T linker_script_file.ld	Use linker_script_file.ld as the linker script. This script replaces the default linker script (rather than adding to it)
-map	Produce a map file
-keepmap	Controls the retention of the map file in the event of a link error
-Mn	Generates a listing of symbols sorted alphabetically/numerically by address
-delete	Instructs the linker to remove functions that are not referenced in the final executable. The linker iterates to find functions that do not have relocations pointing to them and eliminates them
-ignore_debug_references	Ignores relocations from DWARF debug sections when using -delete. DWARF debug information will contain references to deleted functions that may break some third-party debuggers
-Llibrary_path	Points to library_path (the libraries location) for thumb2 to be used for linking
-larch	Link architecture specific library
-lstartup	Link run-time environment startup routines. The source code for themodules in this library is provided in the src/libstartup directory
-lind_sd	Link language-independent library, containing support routines for features such as software floating point, run-time error checking, C99 complex numbers, and some general purpose routines of the ANSI C library
-v	Prints verbose information about the activities of the linker, including the libraries it searches to resolve undefined symbols
-nostartfiles	Controls the start files to be linked into the executable

3.1.3 DIAB Compiler/Assembler/Linker Options

3.1.3.1 DIAB Compiler Options

Compiler Option	Description
-tARMCORTEXM7MG:simple	Selects target processor (hardware single-precision, software double-precision floating-point)
-mthumb	Selects generating code that executes in Thumb state
-std=c99	Follows the C99 standard for C
-Oz	Like -O2 with further optimizations to reduce code size
-g	Generates DWARF 4.0 debug information
-fstandalone-debug	Emits full debug info for all types used by the program
-Wstrict-prototypes	Warn if a function is declared or defined without specifying the argument types
-Wsign-compare	Produce warnings when comparing signed type with unsigned type
-Wdouble-promotion	Give a warning when a value of type float is implicitly promoted to double

Compiler Option	Description
-Wunknown-pragmas	Issues a warning for unknown pragmas
-Wundef	Warns if an undefined identifier is evaluated in an <code>#if</code> directive. Such identifiers are replaced with zero
-Wextra	Enables some extra warning flags that are not enabled by '-Wall'
-Wall	Enables all of the most useful warnings (for historical reasons this option does not literally enable all warnings)
-pedantic	Emits a warning whenever the standard specified by the -std option requires a diagnostic
-Werror=implicit-function-declaration	Generates an error whenever a function is used before being declared
-fno-common	Compile common globals like normal definitions
-fno-signed-char	Char is unsigned
-fno-trigraphs	Do not process trigraph sequences
-V	Displays the current version number of the tool suite
-c	Stop after assembly and produce an object file for each source file
-DS32XX	Predefine S32XX as a macro, with definition 1
-DS32G2XX	Predefine S32G2XX as a macro, with definition 1
-DDIAB	Predefine DIAB as a macro, with definition 1
-DUSE_SW_VECTOR_MODE	Predefine USE_SW_VECTOR_MODE as a macro, with definition 1. By default, the drivers are compiled to handle interrupts in Software Vector Mode
-DD_CACHE_ENABLE	Predefine D_CACHE_ENABLE as a macro, with definition 1. Enables data cache initialization in source file system.c under the Platform driver
-DI_CACHE_ENABLE	Predefine I_CACHE_ENABLE as a macro, with definition 1. Enables instruction cache initialization in source file system.c under the Platform driver
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initialization in source file system.c under the Platform driver
-DMCAL_ENABLE_USER_MODE_SUPPORT	Predefine MCAL_ENABLE_USER_MODE_SUPPORT as a macro, with definition 1. Allows drivers to be configured in user mode

3.1.3.2 DIAB Assembler Options

Assembler Option	Description
-mthumb	Selects generating code that executes in Thumb state
-Xpreprocess-assembly	Invokes C preprocessor on assembly files before running the assembler
-Xassembly-listing	Produces an .lst assembly listing file
-c	Stop after assembly and produce an object file for each source file

3.1.3.3 DIAB Linker Options

Linker Option	Description
-e Reset_Handler	Make the symbol Reset_Handler be treated as a root symbol and the start label of the application
linker_script_file.dld	Use linker_script_file.dld as the linker script. This script replaces the default linker script (rather than adding to it)
-m30	m2 + m4 + m8 + m16
-Xstack-usage	Gathers and display stack usage at link time
-Xpreprocess-lecl	Perform pre-processing on linker scripts
-Llibrary_path	Points to the libraries location for ARMV7EMMG to be used for linking
-lc	Links with the standard C library
-lm	Links with the math library

3.2 Files required for compilation

This section describes the include files required to compile, assemble (if assembler code) and link the Port driver for S32 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.

Port Files

- ..\Port_TS_T40D11M40I0R0\src\Port.c
- ..\Port_TS_T40D11M40I0R0\src\Port_Ipw.c
- ..\Port_TS_T40D11M40I0R0\src\Siul2_Port_Ip.c
- ..\Port_TS_T40D11M40I0R0\include\Port.h
- ..\Port_TS_T40D11M40I0R0\include\Port_Ipw.h
- ..\Port_TS_T40D11M40I0R0\include\Siul2_Port_Ip.h
- ..\Port_TS_T40D11M40I0R0\include\Siul2_Port_Ip_Types.h
- ..\Port_TS_T40D11M40I0R0\include\Siul2_Port_Ip_TrustedFunctions.h

Port Generated Files

- Port_Cfg.c - This file should be generated by the user using a configuration tool for compilation.
- Port_[VariantName]_PBcfg.c - 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.
- Port_[VariantName]_PBcfg.h - This file contains the extern of pointer config structure with PostBuild variant for the respective variant.
- Port_Cfg.h - This file should be generated by the user using a configuration tool for compilation.
- Siul2_Port_Ip_Cfg.c - This file should be generated by the user using a configuration tool for compilation.
- Siul2_Port_Ip_Cfg.h - This file should be generated by the user using a configuration tool for compilation.

Building the driver

- Siul2_Port_Ip_[VariantName]_PBcfg.c - This file should be generated by the user using a configuration tool for compilation.
- Siul2_Port_Ip_[VariantName]_PBcfg.h - This file should be generated by the user using a configuration tool for compilation.
- Siul2_Port_Ip_Defines.h - This file should be generated by the user using a configuration tool for compilation.

As a deviation from standard:

- Port_[VariantName]_PBcfg.c - This files will contain the definition for all parameters that are variant aware, independent of the configuration class that will be selected (PC, LT, PB) for HLD
- Siul2_Port_Ip_[VariantName]_PBcfg.c - This files will contain the definition for all parameters that are variant aware, independent of the configuration class that will be selected (PC, LT, PB) for IPL
- Siul2_Port_Ip_[VariantName]_PBcfg.h - This files will contain the extern of pointer config structure with PostBuild variant for the respective variant.
- Port_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 Port_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_T40D11M40I0R0\include\
- ..\Base_TS_T40D11M40I0R0\header\
- ..\Base_TS_T40D11M40I0R0\src\OsIf_Timer.c
- ..\Base_TS_T40D11M40I0R0\src\OsIf_Timer_System.c

Files from Det folder:

- ..\Det_TS_T40D11M40I0R0\include\Det.h
- ..\Det_TS_T40D11M40I0R0\src\Det.c

Files from Rte folder:

- ..\Rte_TS_T40D11M40I0R0.h
- ..\Rte_TS_T40D11M40I0R0\src\SchM_Port.c

Files from Os folder:

- ..\Os_TS_T40D11M40I0R0.h

Note: <plugin_name>: TS_T<40>D<11>M<SW_Version_Major>I<SW_Version_Minor>R0 (i.e. Target_Id = 40 identifies PowerPC architecture and Derivative_Id = 11 identifies the S32)

3.3 Setting up the plugins

The Port driver was designed to be configured by using the EB Tresos Studio (version EB tresos Studio 27.1.0 or later.)

Location of various files inside the **PORT** module folder:

- VSMD (Vendor Specific Module Definition) file in EB tresos Studio XDM format:
 - `..\Port_TS_T40D11M40I0R0\config\Port.xdm`
- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
 - `Port_TS_T40D11M40I0R0\autosar\Port_s32g233a_bga525.epd`
 - `Port_TS_T40D11M40I0R0\autosar\Port_s32g234m_bga525.epd`
 - `Port_TS_T40D11M40I0R0\autosar\Port_s32g254a_bga525.epd`
 - `Port_TS_T40D11M40I0R0\autosar\Port_s32g274a_bga525.epd`
 - `Port_TS_T40D11M40I0R0\autosar\Port_s32g338m_bga525.epd`
 - `Port_TS_T40D11M40I0R0\autosar\Port_s32g339m_bga525.epd`
 - `Port_TS_T40D11M40I0R0\autosar\Port_s32g358a_bga525.epd`
 - `Port_TS_T40D11M40I0R0\autosar\Port_s32g359a_bga525.epd`
 - `Port_TS_T40D11M40I0R0\autosar\Port_s32g378a_bga525.epd`
 - `Port_TS_T40D11M40I0R0\autosar\Port_s32g379a_bga525.epd`
 - `Port_TS_T40D11M40I0R0\autosar\Port_s32g398a_bga525.epd`
 - `Port_TS_T40D11M40I0R0\autosar\Port_s32g399a_bga525.epd`
- Code Generation Templates for parameters without variation points:
 - `..\Port_TS_T40D11M40I0R0\generate_PC\include\Port_Cfg.h`
 - `..\Port_TS_T40D11M40I0R0\generate_PC\include\Siul2_Port_Ip_Cfg.h`
 - `..\Port_TS_T40D11M40I0R0\generate_PC\include\Siul2_Port_Ip_Defines.h`
 - `..\Port_TS_T40D11M40I0R0\generate_PC\src\Port_Cfg.c`
 - `..\Port_TS_T40D11M40I0R0\generate_PC\src\Siul2_Port_Ip_Cfg.c`
- Code Generation Templates for variant aware parameters:
 - `..\Port_TS_T40D11M40I0R0\generate_PB\include\Port_PBcfg.h`
 - `..\Port_TS_T40D11M40I0R0\generate_PB\include\Siul2_Port_Ip_PBcfg.h`
 - `..\Port_TS_T40D11M40I0R0\generate_PB\src\Port_PBcfg.c`
 - `..\Port_TS_T40D11M40I0R0\generate_PB\src\Siul2_Port_Ip_PBcfg.c`

Steps to generate the configuration:

1. Copy the module folders `Port_TS_T40D11M40I0R0`, `Base_TS_T40D11M40I0R0`, `Resource_TS_T40D11M40I0R0`, `Det_TS_T40D11M40I0R0`, `Rte_TS_T40D11M40I0R0`, `EcuC_TS_T40D11M40I0R0`, `Os_TS_T40D11M40I0R0`, `Platform_TS_T40D11M40I0R0` 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.



Chapter 4

Function calls to module

- [Function Calls during Start-up](#)
- [Function Calls during Shutdown](#)
- [Function Calls during Wake-up](#)

4.1 Function Calls during Start-up

None.

4.2 Function Calls during Shutdown

None.

4.3 Function Calls during Wake-up

None.

Chapter 5

Module requirements

- Exclusive areas to be defined in BSW scheduler
- Exclusive areas not available on this platform
- Peripheral Hardware Requirements
- ISR to configure within AutosarOS - dependencies
- ISR Macro
- Other AUTOSAR modules - dependencies
- Data Cache Restrictions
- User Mode support
- Multicore support

5.1 Exclusive areas to be defined in BSW scheduler

In the current implementation, PORT is using the services of Schedule Manager (SchM) for entering and exiting the critical regions, to preserve a resource. SchM implementation is done by the integrators of the RTD using OS or non-OS services. For testing the PORT, stubs are used for SchM.

Exclusive Areas are used in High level driver layer (HLD)

PORT_EXCLUSIVE_AREA_00 is used in function Port_SetPinMode. It protect the Port_Ipw_au16GpioDirChangeability array resource from read/modify/write operation;

Exclusive Areas are implemented in Low level driver layer (IPL)

PORT_EXCLUSIVE_AREA_01 is used in function Siul2_Port_Ip_SetPullSel. It protect the value of MSCR registers from read/modify/write operation;

PORT_EXCLUSIVE_AREA_02 is used in function Siul2_Port_Ip_SetOutputBuffer. It protect the value of MSCR register from read/modify/write operation;

PORT_EXCLUSIVE_AREA_03 is used in function Siul2_Port_Ip_SetInputBuffer. It protect the value of

Module requirements

MSCR and IMCR registers from read/modify/write operation;

PORT_EXCLUSIVE_AREA_04 is used in function Siul2_Port_Ip_SetPinDirection. It protect the value of MSCR register from read/modify/write operation;

Critical Region Exclusive Matrix

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

#	PORT_EA↔ _00	PORT_EA↔ _01	PORT_EA↔ _02	PORT_EA↔ _03	PORT_EA↔ _04
PORT_EA_00	x				
PORT_EA_01		x	x	x	x
PORT_EA_02		x	x	x	x
PORT_EA_03		x	x	x	x
PORT_EA_04		x	x	x	x

Note

PORT_EA_xx means PORT_EXCLUSIVE_AREA_xx

5.2 Exclusive areas not available on this platform

None.

5.3 Peripheral Hardware Requirements

The PORT driver uses S32's peripheral: SIUL2.

5.4 ISR to configure within AutosarOS - dependencies

None.

5.5 ISR Macro

RTD 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.

5.5.1 Without an Operating System

The macro `_USING_OS_AUTOSAROS_` must not be defined.

5.5.1.1 Using Software Vector Mode

The macro `_USE_SW_VECTOR_MODE_` must be defined and the ISR macro is defined as:

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

In this case, the drivers' interrupt handlers are normal C functions and their prologue/epilogue will handle the context save and restore.

5.5.1.2 Using Hardware Vector Mode

The macro `_USE_SW_VECTOR_MODE_` must not be defined and the ISR macro is defined as:

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

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

5.5.2 With an Operating System Please refer to your OS documentation for description of the ISR macro.

5.6 Other AUTOSAR modules - dependencies

- **DET:** The DET module is used for enabling Development error detection. The API function used is `Det_ReportError()`. The activation / deactivation of Development error detection is configurable using the `PortDevErrorDetect` configuration parameter.
- **BASE:** The BASE module contains the common files/definitions needed by all RTD modules.
- **RESOURCE:** The RESOURCE module is used to select microcontroller derivatives.
- **RTE:** The RTE module is used to manage the exclusive area inside PORT driver
- **ECUC:** The ECUC module is used for ECU configuration. RTD modules need ECUC to retrieve the variant information.
- **OS:** The OS module is used for OS configuration. RTD modules need OS to retrieve the application information.
- **MCU:** The MCU driver provides services for basic microcontroller initialization, power down functionality, reset and microcontroller specific functions required by other RTD software modules. The clocks need to be initialized prior to using the PORT driver.

5.7 Data Cache Restrictions

None.

5.8 User Mode support

- [User Mode configuration in the module](#)
- [User Mode configuration in AutosarOS](#)

5.8.1 User Mode configuration in the module

The Port module can be run from user mode if 'PortEnableUserModeSupport' is enabled in the configuration.

In this case, the Port module will set the UAA bit in REG_PROT_GCR register of SIUL2_0 HW IP.

In addition to setting 'PortEnableUserModeSupport' in the configuration, the application shall call the following functions as trusted functions:

Function syntax	Description	Available via
Siul2_Port_Ip_SetUserAccess↔ Allowed(void)	Setting the user access allowed for SIUL2 registers protected by REG↔ _PROT	Siul2_Port_Ip_Trusted↔ Functions.h

5.8.2 User Mode configuration in AutosarOS

When User mode is enabled, the driver may has the functions that need to be called as trusted functions in AutosarOS context. Those functions are already defined in driver and declared in the header <IpName>_Ip↔_TrustedFunctions.h. This header also included all headers files that contains all types definition used by parameters or return types of those functions. Refer the chapter [User Mode configuration in the module](#) for more detail about those functions and the name of header files they are declared inside. Those functions will be called indirectly with the naming convention below in order to AutosarOS can call them as trusted functions.

```
Call_<Function_Name>_TRUSTED(parameter1,parameter2,...)
```

That is the result of macro expansion `OsIf_Trusted_Call` in driver code:

```
#define OsIf_Trusted_Call[1-6params](name,param1,...,param6) Call_##name##_TRUSTED(param1,...,param6)
```

So, the following steps need to be done in AutosarOS:

- Ensure `MCAL_ENABLE_USER_MODE_SUPPORT` macro is defined in the build system or somewhere global.
- Define and declare all functions that need to call as trusted functions follow the naming convention above in Integration/User code. They need to visible in `Os.h` for the driver to call them. They will do the marshalling of the parameters and call `CallTrustedFunction()` in OS specific manner.
- `CallTrustedFunction()` will switch to privileged mode and call `TRUSTED_<Function_Name>()`.
- `TRUSTED_<Function_Name>()` function is also defined and declared in Integration/User code. It will un-marshalling of the parameters to call `<Function_Name>()` of driver. The `<Function_Name>()` functions are already defined in driver and declared in `<IpName>_Ip_TrustedFunctions.h`. This header should be included in OS for OS call and indexing these functions.

See the sequence chart below for an example calling `Linflexd_Uart_Ip_Init_Privileged()` as a trusted function.

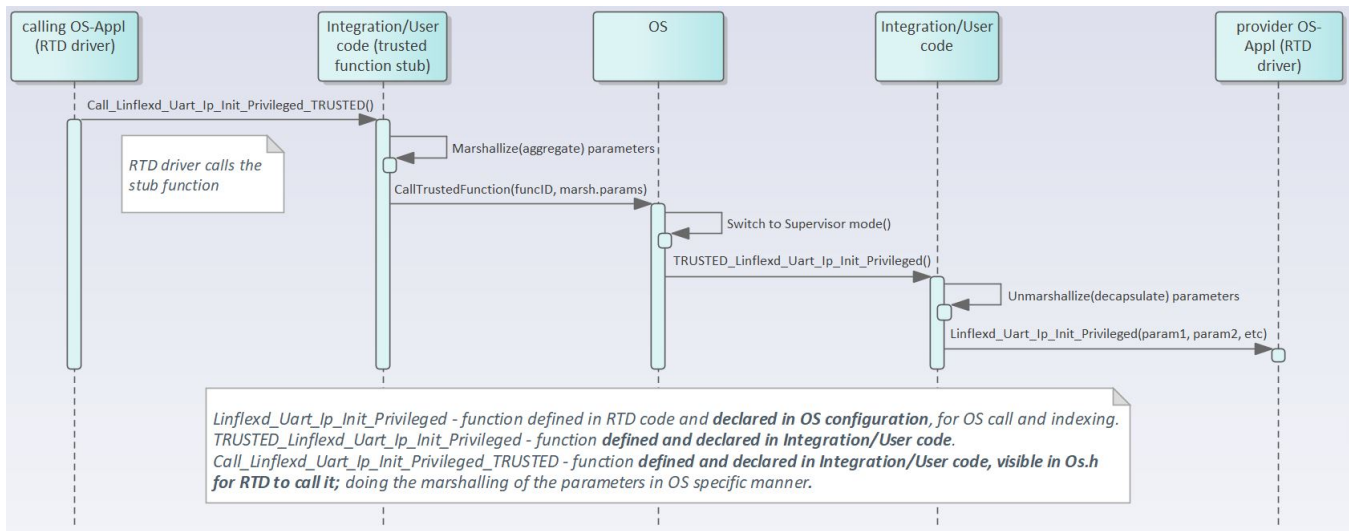


Figure 5.1 Example sequence chart for calling `Linflexd_Uart_Ip_Init_Privileged` as trusted function

5.9 Multicore support

The **Port** implements the "Autosar 4.4 MCAL Multicore Distribution" according to type III, in which the mappable element is set to Pins. For additional details, please refer to **AUTOSAR_EXP_BSWDistribution**↵
Guide.

The **Port** and the mappable elements can be allocated to zero, one or several ECUC partitions, by means of "↵
PortEcucPartionRef". If the Port is mapped to zero ECUC partitions, the Port behavior reverts to single-core implementation, similar to previous Autosar versions. If the Port is mapped to one or more ECUC partitions, the Port enforces the following multi-core assumptions:

1. The Port assumes there is a single EcucPartition allocated per core. Internally, the module will use the Core ID returned by GetCoreID API to reference the appropriate global data and configuration elements.
2. The Port assumes the EcucCoreIDs are defined in a compact/consecutive order, starting from zero. The rationale is that the number of EcucPartitions is used for dimensioning the Port internal variables and the EcucCoreIDs are used for indexing those variables.(AR-86601 Zero based and dense IDs for OS-Cores and OS-Applications)
3. The Port assumes that initialization is performed on a single, designated core. Port_Init() is called only once, with a single configuration structure. (Type III).
4. The Port initialization expects the upper layer will pass the correct initialization pointer, specific to the partition in which the driver is to be used. For example: EcucPartition_1 is assigned to CoreID 1; Port_Init function will be called with configuration structure init, on Core 1.
5. The Port will check upon each API call if the requested resource is configured to be available on the current core, if DET error reporting is enabled.

Module requirements

6. The Port requires that all variables in NonCacheable MemMap sections be allocated accordingly, to avoid data corruption in multicore context.
7. The Port assumes that RTE module implements the EXCLUSIVE AREAS to be coreaware only. The rationale is that the module implementation ensures data integrity by separating the mappable elements for different cores already, thus implementing the EXCLUSIVE AREAS in a blocking manner (ex: spin-lock) on a multicore scope, might affect the performance of the drivers on the two cores, although they might access separate HW elements. For single-core scope, the EXCLUSIVE AREAS keep the same purpose as on previous AUTOSAR implementations. (to be updated per Port usecase, to be detailed/removed if some modules require such kind of functionality for critical features which cannot be atomically shared among cores).



Chapter 6

Main API Requirements

- [Main function calls within BSW scheduler](#)
- [API Requirements](#)
- [Calls to Notification Functions, Callbacks, Callouts](#)

6.1 Main function calls within BSW scheduler

None.

6.2 API Requirements

None.

6.3 Calls to Notification Functions, Callbacks, Callouts

None.

Chapter 7

Memory allocation

- [Sections to be defined in Port_MemMap.h](#)
- [Linker command file](#)

7.1 Sections to be defined in Port_MemMap.h

Section name	Section type	Description
PORT_START_SEC_CODE	Code	Start of Memory Section for Code
PORT_STOP_SEC_CODE	Code	End of Memory Section for Code
PORT_START_SEC_CONFIG_DATA↔ _UNSPECIFIED	Configuration Data	Start of Memory Section for Config Data
PORT_STOP_SEC_CONFIG_DATA↔ _UNSPECIFIED	Configuration	Data End of Memory Section for Config Data
PORT_START_SEC_CONFIG_DATA↔ _8	Variables	Start of Memory Section for Config Data
PORT_STOP_SEC_CONFIG_DATA_8	Variables	Stop of Memory Section for Config Data
PORT_START_SEC_CONFIG_DATA↔ _16	Variables	Start of Memory Section for Config Data
PORT_STOP_SEC_CONFIG_DATA_16	Variables	Stop of Memory Section for Config Data
PORT_START_SEC_CONFIG_DATA↔ _32	Variables	Start of Memory Section for Config Data
PORT_STOP_SEC_CONFIG_DATA_32	Variables	Stop of Memory Section for Config Data
PORT_START_SEC_VAR_CLEARED↔ _UNSPECIFIED_NO_CACHEABLE	Variables	Variables are cleared to zero by start-up code and located in shared data region
PORT_STOP_SEC_VAR_CLEARED↔ _UNSPECIFIED_NO_CACHEABLE	Variables	End of above section.
PORT_START_SEC_VAR_CLEARED↔ _16_NO_CACHEABLE	Variables	Variables are cleared to zero by start-up code and located in shared data region
PORT_STOP_SEC_VAR_CLEARED↔ _16_NO_CACHEABLE	Variables	End of above section.
PORT_START_SEC_CONST_32	Variables	The parameters that are not variant aware shall be stored in memory section for constants.

Section name	Section type	Description
PORT_STOP_SEC_CONST_32	Variables	End of above section.
PORT_START_SEC_CONST_16	Variables	The parameters that are not variant aware shall be stored in memory section for constants.
PORT_STOP_SEC_CONST_16	Variables	End of above section.
PORT_START_SEC_CONST_↵ UNSPECIFIED	Variables	The parameters that are not variant aware shall be stored in memory section for constants.
PORT_STOP_SEC_CONST_↵ UNSPECIFIED	Variables	End of above section.

7.2 Linker command file

Memory shall be allocated for every section defined in the driver's "<Module>_MemMap.h.



Chapter 8

Integration Steps

This section gives a brief overview of the steps needed for integrating this module:

1. Generate the required module configuration(s). For more details refer to section [Files Required for Compilation](#)
2. Allocate the proper memory sections in the driver's memory map header file ("`<Module>_MemMap.h`") and linker command file. For more details refer to section [Sections to be defined in `<Module>_MemMap.h`](#)
3. Compile & build the module with all the dependent modules. For more details refer to section [Building the Driver](#)

Chapter 9

External assumptions for driver

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

External Assumption Req ID	External Assumption Text
SWS_Port_00006	The user of the PORT Driver module shall configure the symbolic names of the port pins of the MCU. Note: Out of scope
SWS_Port_00078	The Port Driver module's environment shall call the function Port_Init first in order to initialize the port for use. Note: Out of scope
SWS_Port_00213	If Port_Init function is not called first, then no operation can occur on the MCU ports and port pins. Note: Out of scope
SWS_Port_00215	If the register can affect several hardware modules and if it is not an I/O register, it shall be initialised by the MCU driver. Note: Reason: These requirements are not related to PORT
SWS_Port_00217	One-time writable registers that require initialisation directly after reset shall be initialised by the startup code. Note: Reason: These requirements are not related to PORT
SWS_Port_00218	All the other registers not mentioned before, shall be initialised by the start-up code. Note: Reason: These requirements are not related to PORT
SWS_Port_00071	The Port Driver module's environment shall call the function Port_Init after a reset in order to reconfigure the ports and port pins of the MCU. Note: Out of scope sMcal
EA_RTD_00051	The application shall ensure that Port_Init() is not preempting itself or other PORT functions.
EA_RTD_00052	The application shall ensure that Port_SetPinDirection() and Port_SetPinMode() are not preempting themselves or one each other when called on the same port.
EA_RTD_00071	If interrupts are locked, a centralized function pair to lock and unlock interrupts shall be used.
EA_RTD_00082	When caches are enabled and data buffers are allocated in cacheable memory regions the buffers involved in DMA transfer shall be aligned with both start and end to cache line size. Note: Rationale: This ensures that no other buffers/variables compete for the same cache lines.
EA_RTD_00092	The integrator shall allocate a single EcucPartition per core or the partition in which the Port is allocated shall be exclusively mapped to a core. Note: Internally, the Port will use the Core ID returned by GetCoreID API to reference the appropriate global data and configuration elements, that is why a core should reference only one configured partition.

External assumptions for driver

External Assumption Req ID	External Assumption Text
EA_RTD_00093	The application shall define EcucCoreIDs in a compact/consecutive order, starting from zero.
EA_RTD_00095	The application shall call Port_Init() on a single, designated core, using a single configuration pointer.
EA_RTD_00096	The application shall pass the correct initialization pointer, specific to the partition in which the driver is to be used.
EA_RTD_00106	Standalone IP configuration and HL configuration of the same driver shall be done in the same project
EA_RTD_00107	The integrator shall use the IP interface only for hardware resources that were configured for standalone IP usage. Note: The integrator shall not directly use the IP interface for hardware resources that were allocated to be used in HL context.
EA_RTD_00108	The integrator shall use the IP interface to build a CDD, therefore the BSWMD will not contain reference to the IP interface

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.

© 2022 NXP B.V.

