Integration Manual

for S32K3 LIN Driver

Document Number: IM34LINASRR21-11 Rev0000R3.0.0 Rev. 1.0

Chapter 1

Revision History

Revision	Date	Author	Description
1.0	31.03.2023	NXP RTD Team	S32K3 Real-Time Drivers AUTOSAR 4.4 & R21-11 Version 3.0.0

Chapter 2

Introduction

This integration manual describes the integration requirements for LIN Driver for S32K3XX microcontrollers.

2.1 Supported Derivatives

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

- s32k310_mqfp100
- s32k310_lqfp48
- s32k311_mqfp100 / MWCT2015S_mqfp100
- s32k311_lqfp48
- s32k312_mqfp100 / MWCT2016S_mqfp100
- s32k312_mqfp172 / MWCT2016S_mqfp172
- s32k314_mqfp172
- s32k314_mapbga257
- s32k322_mqfp100 / MWCT2D16S_mqfp100
- s32k322_mqfp172 / MWCT2D16S_mqfp172
- s32k324_mqfp172 / MWCT2D17S_mqfp172
- s32k324_mapbga257

- s32k341_mqfp100
- s32k341_mqfp172
- s32k342_mqfp100
- s32k342_mqfp172
- s32k344_mqfp172
- s32k344_mapbga257
- s32k394_mapbga289
- s32k396_mapbga289
- s32k358_mqfp172
- s32k358_mapbga289
- s32k328_mqfp172
- s32k328_mapbga289
- s32k338_mqfp172
- s32k338_mapbga289
- s32k348_mqfp172
- s32k348_mapbga289
- s32m274_lqfp64
- s32m276_lqfp64

All of the above microcontroller devices are collectively named as S32K3.

Note: MWCT part numbers contain NXP confidential IP for Qi Wireless Power.

2.2 Overview

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

AUTOSAR:

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

2.3 About This Manual

This Technical Reference employs the following typographical conventions:

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

Notes and warnings are shown as below:

Note

This is a note.

Warning

This is a warning

2.4 Acronyms and Definitions

Term	Definition
API	Application Programming Interface
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 Signifigant 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

Term	Definition
API	Application Programming Interface
ASM	Assembler
BSMI	Basic Software Make file Interface
C/CPP	C and C++ Source Code
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECUM	ECU state Manager
ECU	Electronic Control Unit
ISR	Interrupt Service Routine
LIN	Local Interconnect Network
LSB	Least Signifigant Bit
MCU	Micro Controller Unit
MSB	Most Significant Bit
N/A	Not Applicable
RAM	Random Access Memory
SIU	Systems Integration Unit
SWS	Software Specification

Term	Definition
VLE	Variable Length Encoding
XML	Extensible Markup Language

2.5 Reference List

#	Title	Version
1	Specification of LIN Driver	AUTOSAR Release R21-11
2	Reference Manual	S32K3xx Reference Manual, Rev.6, Draft B, 01/2023
3	Reference Manual for S32K39 and S32K37	S32K39 and S32K37 Reference Manual, Rev. 2 Draft A, 11/2022
4	Reference Manual for S32M27x	S32M27x Reference Manual, Rev.2, Draft A, — 02/2023
5	Datasheet	S32K3xx Data Sheet, Rev. 6, 11/2022
6	Datasheet for S32K39 and S32K37	S32K396 Data Sheet, Rev. 1.1 — 08/2022
7	Datasheet for S32M27x	S32M2xx Data Sheet, Rev. 2 RC — 12/2022
8	Errata	S32K358_0P14E Mask Set Errata – Rev. 28, 9/2022
		S32K396_0P40E Mask Set Errata, Rev. DEC2022, 12/2022
		S32K311_0P98C Mask Set Errata, Rev. 6/March/2023, 3/2023
		S32K312: Mask Set Errata for Mask 0P09C, Rev. 25/↩ April/2022
		S32K342: Mask Set Errata for Mask 0P97C, Rev. 10, 11/2022
		S32K3x4: Mask Set Errata for Mask 0P55A/1P55A, Rev. 14/Oct/2022

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
- DIAB Compiler/Assembler/Linker Options
- GHS Compiler/Assembler/Linker Options
- IAR Compiler/Assembler/Linker Options

The RTD driver files are compiled using:

- NXP GCC 10.2.0 20200723 (Build 1728 Revision g5963bc8)
- Wind River Diab Compiler 7.0.4
- Compiler Versions: Green Hills Multi 7.1.6d / Compiler 2021.1.4
- Compiler Versions: IAR ANSI C/C++ Compiler V8.50.10 (safety version)

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

The TS_T40D34M30I0R0 part of the plugin name is composed as follows:

- T = Target_Id (e.g. T40 identifies Cortex-M architecture)
- D = Derivative_Id (e.g. D34 identifies S32K3 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
-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
-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 aforementioend -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

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
-DS32K3XX	Predefine S32K3XX as a macro, with definition 1
-D $ (DERIVATIVE)	Predefine S32K3's derivative as a macro, with definition 1. For example: Predefine for S32K344 will be -DS32K344.
-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 initalization 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 initalization in source file system.c under the Platform driver
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initalization in source file system.c under the Platform driver
-DMCAL_ENABLE_USER_MODE_SUPPORT	Predefine MCAL_ENABLE_USER_MODE_SUPPO↩ RT as a macro, with definition 1. Allows drivers to be configured in user mode.
–sysroot=	Specifies the path to the sysroot, for Cortex-M7 it is /arm-none-eabi/newlib
-specs=nano.specs	Use Newlib nano specs
-specs=nosys.specs	Do not use printf/scanf

3.1.1.2 GCC Assembler Options

Assembler Option	Description
-Xassembler-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

Assembler Option	Description
-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
-specs=nano.specs	Use Newlib nano specs
-specs=nosys.specs	Do not use printf/scanf

3.1.2 DIAB Compiler/Assembler/Linker Options

3.1.2.1 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 #if 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
-DS32K3XX	Predefine S32K3XX as a macro, with definition 1
-D $ (DERIVATIVE)	Predefine S32K3's derivative 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 initalization 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 initalization in source file system.c under the Platform driver
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initalization in source file system.c under the Platform driver
-DMCAL_ENABLE_USER_MODE_SUPPORT	Predefine MCAL_ENABLE_USER_MODE_SUPPO↩ RT as a macro, with definition 1. Allows drivers to be configured in user mode

3.1.2.2 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
-tARMCORTEXM7MG:simple	Selects target processor (hardware single-precision, software double-precision floating-point)

3.1.2.3 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
-tARMCORTEXM7MG:simple	Selects target processor (hardware single-precision, software double-precision floating-point)

3.1.3 GHS Compiler/Assembler/Linker Options

3.1.3.1 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
-Wtrigraphs	Produce warnings if trigraphs are detected
-Wundef	Produce a warning if undefined identifiers are used in #if preprocessor statements

Compiler Option	Description
–unsigned_chars	Let the type char be unsigned, like unsigned char
–unsigned_fields	Bitfelds 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 andgenerate 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
-DS32K3XX	Predefine S32K3XX as a macro, with definition 1
-D $ (DERIVATIVE)	Predefine S32K3's derivative as a macro, with definition 1. For example: Predefine for S32K344 will be -DS32K344.
-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 initalization 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 initalization in source file system.c under the Platform driver
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initalization in source file system.c under the Platform driver
-DMCAL_ENABLE_USER_MODE_SUPPORT	Predefine MCAL_ENABLE_USER_MODE_SUPPO↩ RT as a macro, with definition 1. Allows drivers to be configured in user mode

3.1.3.2 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

3.1.3.3 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. DWA↩ RF 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
-keep=C40_Ip_AccessCode	Avoid linker remove function C40_Ip_AccessCode from Fls module because it is not referenced explicitly
-nostartfiles	Controls the start files to be linked into the executable

3.1.4 IAR Compiler/Assembler/Linker Options

3.1.4.1 IAR Compiler Options

Compiler Option	Description
–cpu Cortex-M7	Targeted ARM processor for which IAR should tune the performance of the code
–cpu_mode thumb	Generates code that executes in Thumb state
–endian little	Generate code for a processor running in little-endian mode
–fpu VFPv5-SP	Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). Single-precision variant.
-e	Enables all IAR C language extensions
-Ohz	Optimize for size. the compiler will emit AEABI attributes indicating the requested optimization goal. This information can be used by the linker to select smaller or faster variants of DLIB library functions
–debug	Makes the compiler include debugging information in the object modules. Including debug information will make the object files larger

Compiler Option	Description
–no_clustering	Disables static clustering optimizations. Static and global variables defined within the same module will not be arranged so that variables that are accessed in the same function are close to each other
–no_mem_idioms	Makes the compiler not optimize certain memory access patterns
–do_explicit_zero_opt_in_named_sections	Disable the exception for variables in user-named sections, and thus treat explicit initializations to zero as zero initializations, not copy initializations
–require_prototypes	Force the compiler to verify that all functions have proper prototypes. Generates an error otherwise
–no_wrap_diagnostics	Does not wrap long lines in diagnostic messages
–diag_suppress Pa050	Suppresses diagnostic message Pa050
-DS32K3XX	Predefine S32K3XX as a macro, with definition 1
-D $ (DERIVATIVE)	Predefine S32K3's derivative as a macro, with definition 1. For example: Predefine for S32K344 will be -DS32K344.
-DIAR	Predefine IAR 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 initalization 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 initalization in source file system.c under the Platform driver
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initalization in source file system.c under the Platform driver
-DMCAL_ENABLE_USER_MODE_SUPPORT	Predefine MCAL_ENABLE_USER_MODE_SUPPO↩ RT as a macro, with definition 1. Allows drivers to be configured in user mode.

3.1.4.2 IAR Assembler Options

Assembler Option	Description
–cpu Cortex-M7	Targeted ARM processor for which IAR should generate the instruction set
–fpu VFPv5-SP	Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). Single-precision variant.
–cpu_mode thumb	Selects the thumb mode for the assembler directive CODE
-g	Disables the automatic search for system include files
-r	Generates debug information

3.1.4.3 IAR Linker Options

Linker Option	Description
–map filename	Produces a map file
–config linkerfile	Use linkerfile as the linker script. This script replaces the default linker script (rather than adding to it)
–cpu=Cortex-M7	Selects the ARM processor variant to link the application for
–fpu VFPv5-SP	Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). Single-precision variant.
–entry _start	Treats _start as a root symbol and start label
–enable_stack_usage	Enables stack usage analysis. If a linker map file is produced, a stack usage chapter is included in the map file
–skip_dynamic_initialization	Dynamic initialization (typically initialization of C++ objects with static storage duration) will not be performed automatically during application startup
–no_wrap_diagnostics	Does not wrap long lines in diagnostic messages

3.2 Files required for compilation

This section describes the include files required to compile, assemble (if assembler code) and link the LIN driver for S32K3XX 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.

3.2.1 LIN Driver Files

- Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\src\Lin_43_LPUART_FLEXIO.c
- Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\src\Lin_43_LPUART_FLEXIO_ASRExt.c
- Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\src\Lin_43_LPUART_FLEXIO_Ipw.c
- Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\src\Lin_Ip_Common.c
- Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\src\Lpuart_Lin_Ip_Hw_Access.c
- Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\src\Lpuart_Lin_Ip_Irq.c
- Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\src\Lpuart_Lin_Ip.c
- Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\src\Flexio_Lin_Ip.c
- Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\include\Lin_43_LPUART_FLEXIO.h
- Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\include\Lin_43_LPUART_FLEXIO_ASRExt.h
- Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\include\Lin_43_LPUART_FLEXIO_Types.h
- Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\include\Lin_43_LPUART_FLEXIO_Ipw.h
- Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\include\Lin_43_LPUART_FLEXIO_Ipw_Types.h
- Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\include\Lin_Ip_Common.h
- Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\include\Lpuart_Lin_Ip_Hw_Access.h

- Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\include\Lpuart_Lin_Ip.h
- Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\include\Lpuart_Lin_Ip_Types.h
- Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\include\Flexio_Lin_Ip_Irq.h
- Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\include\Flexio_Lin_Ip_Types.h
- Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\include\Flexio_Lin_Ip.h

3.2.2 LIN Driver Generated Files (must be generated by the user using a configuration tool):

- Lin_43_LPUART_FLEXIO_Cfg.c
- Lin_43_LPUART_FLEXIO_Cfg.h
- Lin_43_LPUART_FLEXIO_Defines.h
- Lin_43_LPUART_FLEXIO_Ipw_Cfg.h
- Lpuart_Lin_Ip_Cfg.h
- Lpuart_Lin_Ip_Defines.h
- Flexio_Lin_Ip_CfgDefines.h
- Flexio_Lin_Ip_Defines.h
- Flexio_Lin_Ip_Cfg.h
- Lin_43_LPUART_FLEXIO_[VariantName]_PBcfg.c
- Lin_43_LPUART_FLEXIO_Ipw_[VariantName]_PBcfg.c
- Lpuart_Lin_Ip_[VariantName]_PBcfg.c
- Flexio_Lin_Ip_[VariantName]_PBcfg.c
- Lin_43_LPUART_FLEXIO_[VariantName]_PBcfg.h
- Lin_43_LPUART_FLEXIO_Ipw_[VariantName]_PBcfg.h
- Lpuart_Lin_Ip_[VariantName]_PBcfg.h
- Flexio_Lin_Ip_[VariantName]_PBcfg.h

3.2.3 BASE Files:

- BaseNXP_TS_T40D34M30I0R0\include\Lin_GeneralTypes.h
- BaseNXP_TS_T40D34M30I0R0\include\Mcal.h
- BaseNXP_TS_T40D34M30I0R0\include\Lin_43_LPUART_FLEXIO_MemMap.h
- BaseNXP_TS_T40D34M30I0R0\include\Platform_Types.h
- BaseNXP_TS_T40D34M30I0R0\include\Soc_Ips.h
- BaseNXP_TS_T40D34M30I0R0\include\StandardTypes.h

- BaseNXP_TS_T40D34M30I0R0\include\OsIf.h
- BaseNXP_TS_T40D34M30I0R0\include\Devassert.h
- BaseNXP_TS_T40D34M30I0R0\generate_PC\include\modules.h
- BaseNXP_TS_T40D34M30I0R0\header\[PlatformName]_LPUART.h
- BaseNXP_TS_T40D34M30I0R0\header\[PlatformName]_FLEXIO.h

3.2.4 DEM Files:

- Dem_TS_T40D34M30I0R0\include\Dem.h
- Dem_TS_T40D34M30I0R0\include\Dem_Types.h
- Dem_TS_T40D34M30I0R0\generate_PC\include\Dem_IntErrId.h
- Dem_TS_T40D34M30I0R0\src\Dem.c

3.2.5 DET Files:

- Det_TS_T40D34M30I0R0\include\Det.h
- Det_TS_T40D34M30I0R0\src\Det.c

3.2.6 RTE Files:

- Rte_TS_T40D34M30I0R0\include\SchM_Lin_43_LPUART_FLEXIO.h
- Rte_TS_T40D34M30I0R0\src\SchM_Lin_43_LPUART_FLEXIO.c

3.2.7 LINIF Files:

- LinIf_TS_T40D34M30I0R0\include\LinIf.h
- LinIf_TS_T40D34M30I0R0\src\LinIf.c

3.2.8 ECUM folder:

- EcuM_TS_T40D34M30I0R0\generate_PC\include\EcuM_Cfg.h
- EcuM_TS_T40D34M30I0R0\include\EcuM.h
- EcuM_TS_T40D34M30I0R0\src\EcuM.c

3.2.9 MCL folder:

- Mcl_TS_T40D34M30I0R0\generate_PC\include\Flexio_Mcl_Ip_Cfg.h
- Mcl_TS_T40D34M30I0R0\generate_PC\include\Flexio_Mcl_Ip_CfgDefines.h
- Mcl_TS_T40D34M30I0R0\generate_PC\include\Flexio_Mcl_Ip_Definitions.h
- Mcl_TS_T40D34M30I0R0\generate_PB\include\Flexio_Mcl_Ip_PBcfg.h
- Mcl_TS_T40D34M30I0R0\generate_PB\include\Flexio_Mcl_Ip_PBcfg.c
- Mcl_TS_T40D34M30I0R0\include\Mcl.h
- Mcl_TS_T40D34M30I0R0\include\Flexio_Mcl_Ip_HwAccess.h
- Mcl_TS_T40D34M30I0R0\include\Flexio_Mcl_Ip_Types.h
- Mcl_TS_T40D34M30I0R0\include\Flexio_Mcl_Ip.h
- Mcl_TS_T40D34M30I0R0\src\Mcl.c
- Mcl_TS_T40D34M30I0R0\src\Flexio_Mcl_Ip_HwAccess.c
- Mcl_TS_T40D34M30I0R0\src\Flexio_Mcl_Ip_Types.c
- Mcl_TS_T40D34M30I0R0\src\Flexio_Mcl_Ip.c
- Mcl_TS_T40D34M30I0R0\src\Flexio_Mcl_Ip_Irq.c

3.3 Setting up the plugins

3.3.1 Location of various files inside the Lin module folder:

- VSMD (Vendor Specific Module Definition) file in EB Tresos Studio XDM format:
  - Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\config\Lin_43_LPUART_FLEXIO.xdm
- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\autosar\Lin_43_LPUART_FLEXIO_<subderivative↩ _name>.epd
- Code Generation Templates for variant aware parameters:
  - Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\generate_PB\src\Lin_43_LPUART_FLEXIO↩ _Ipw_PBcfg.c
  - Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\generate_PB\src\Lin_43_LPUART_FLEXIO↩ _PBcfg.c
  - Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\generate_PB\src\Lpuart_Lin_Ip_PBcfg.c
  - Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\generate_PB\src\Flexio_Lin_Ip_PBcfg.c
  - Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\generate_PB\include\Lin_43_LPUART_FLE↩ XIO_Ipw_PBcfg.h
  - Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\generate_PB\include\Lin_43_LPUART_FLE↩ XIO_PBcfg.h

  - Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\generate_PB\include\Lpuart_Lin_Ip_PBcfg.h
  - Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\generate_PB\include\Flexio_Lin_Ip_PBcfg.h

- Code Generation Templates for parameters without variation points:
  - Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\generate_PC\include\Lin_43_LPUART_FLE↩ XIO_Cfg.h
  - Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\generate_PC\include\Lin_43_LPUART_FLE↩ XIO_Defines.h
  - Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\generate_PC\include\Lin_43_LPUART_FLE↩ XIO_Ipw_Cfg.h
  - Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\generate_PC\include\Lpuart_Lin_Ip_Cfg.h
  - Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\generate_PC\include\Lpuart_Lin_Ip_Defines.h
  - Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\generate_PC\include\Flexio_Lin_Ip_Cfg↩ Defines.h
  - Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\generate_PC\include\Flexio_Lin_Ip_Defines.h
  - Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\generate_PC\include\Flexio_Lin_Ip_Cfg.h
  - Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0\generate_PC\src\Lin_43_LPUART_FLEXIO↩ _Cfg.h

3.3.2 Steps to generate the configuration:

1. Copy the following module folders into the Tresos plugins folder:
   - Lin_43_LPUART_FLEXIO_TS_T40D34M30I0R0
   - BaseNXP_TS_T40D34M30I0R0
   - Dem_TS_T40D34M30I0R0
   - Det_TS_T40D34M30I0R0
   - EcuC_TS_T40D34M30I0R0
   - EcuM_TS_T40D34M30I0R0
   - Rte_TS_T40D34M30I0R0
   - Resource_TS_T40D34M30I0R0
   - Mcu_TS_T40D34M30I0R0
   - LinIf_TS_T40D34M30I0R0
   - Mcl_TS_T40D34M30I0R0
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

4.1 Function Calls during Start-up

LIN shall be initialized during STARTUP phase of EcuM initialization. The API to be called for this is Lin_43_↩
LPUART_FLEXIO_Init(). The MCU module should be initialized before LIN module initialization.

The Lin driver does not need OS Support except for ISR's. Hence, can be initialized either in STARTUP1 or S↩
TARTUP2 phase of EcuM initialization. This depends on the implementation, desired duration for STARTUP1 & Target hardware design.

The LIN module shall be initialized by Lin_43_LPUART_FLEXIO_Init(<&Lin_Configuration>) service call during the start-up before the LIN peripherals are used.

Please note that GPIO pins used for connection of LIN physical layer have to be properly assigned to desired L↩
PUART module prior the LIN initialization: so the MCU and PORT modules shall be initialized before LIN is initialized. After the LIN module is initialized each LIN channel has to be initialized as well before using it. This is also done by the Lin_43_LPUART_FLEXIO_Init(<&Lin_Configuration>) service.

In case FLEXIO channel is used, Mcl_Init(<&Mcl_configuration>) must be called before the lin driver init.

4.2 Function Calls during Shutdown

There is no shutdown specific procedure for Lin driver. LIN driver can go to sleep mode using the following API services:

- Lin_43_LPUART_FLEXIO_GoToSleepInternal(LIN_CHANNEL): which put the LIN driver into sleep mode without sending of Go-to-sleep command over the bus.
- Lin_43_LPUART_FLEXIO_GoToSleep(LIN_CHANNEL): which put the LIN driver into sleep mode and send

4.3 Function Calls during Wake-up

LIN driver supports the transmission of wake up command via the LIN bus.

For this purposes the Lin_43_LPUART_FLEXIO_Wakeup(LIN_CHANNEL) API service may be used.

4.3.1 External Wakeup:

The Lin driver supports external wake-up from the bus in 2 modes:

Upon wakeup detection on Rx pin of the configured LPUART channel the ISR "LIN_LPUARTx_RxTx_IRQ↩ Handler()" will be executed (based on the LIN Channel configured) to wake-up Lin Driver.

Upon wakeup detection on Rx pin of the configured FLEXIO channel the ISR "Flexio_Lin_Ip_IrqHandler()" will be executed (based on the LIN Channel configured) to wake-up Lin Driver.

Chapter 5

Module requirements

5.1 Exclusive areas to be defined in BSW scheduler

In the current implementation, LIN 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 MCAL using OS or non-OS services. For testing the LIN, stubs are used for SchM. The following critical regions are used in the LIN driver: **Exclusive Areas are used in High level driver layer (HLD)**

- **LIN_EXCLUSIVE_AREA_08** is used in function Lin_43_LPUART_FLEXIO_Init to protect the C↩ TRL register from the write operation in Lpuart_Lin_Ip_Init.
- **LIN_EXCLUSIVE_AREA_09** is used in function Lin_43_LPUART_FLEXIO_Init to protect the B↩ AUD, CTRL registers from the write operation in Lpuart_Lin_Ip_Init.
- **LIN_EXCLUSIVE_AREA_09** is used in function Lin_43_LPUART_FLEXIO_SendFrame to protect the for BAUD, CTRL registers from the write operation in Lpuart_Lin_Ip_AbortTransferData.
- **LIN_EXCLUSIVE_AREA_09** is used in function Lin_43_LPUART_FLEXIO_GoToSleep to protect the for BAUD, CTRL registers from the write operation in Lpuart_Lin_Ip_AbortTransferData.

- **LIN_EXCLUSIVE_AREA_09** is used in function Lin_43_LPUART_FLEXIO_GoToSleepInternal to protect the for BAUD, CTRL registers from the write operation in Lpuart_Lin_Ip_AbortTransferData.
- **LIN_EXCLUSIVE_AREA_09** is used in function Lin_43_LPUART_FLEXIO_Ipw_Callback to protect the for BAUD, CTRL registers from the write operation in Lpuart_Lin_Ip_AbortTransferData.
- **LIN_EXCLUSIVE_AREA_09** is used in function Lin_43_LPUART_FLEXIO_Init to protect the for BAUD, CTRL registers from the write operation in Lpuart_Lin_Ip_GoToSleepMode.
- **LIN_EXCLUSIVE_AREA_09** is used in function Lin_43_LPUART_FLEXIO_GoToSleepInternal to protect the for BAUD, CTRL registers from the write operation in Lpuart_Lin_Ip_GoToSleepMode.
- **LIN_EXCLUSIVE_AREA_09** is used in function Lin_43_LPUART_FLEXIO_Ipw_Callback to protect the for BAUD, CTRL registers from the write operation in Lpuart_Lin_Ip_GoToSleepMode.
- **LIN_EXCLUSIVE_AREA_09** is used in function Lin_43_LPUART_FLEXIO_SendFrame to protect the BAUD register from the write operation in Lpuart_Lin_Ip_SendFrame.
- **LIN_EXCLUSIVE_AREA_09** is used in function Lin_43_LPUART_FLEXIO_WakeUpInternal to protect the for BAUD, CTRL registers from the write operation in Lpuart_Lin_Ip_GotoIdleState.
- **LIN_EXCLUSIVE_AREA_09** is used in function Lin_43_LPUART_FLEXIO_SendFrame to protect the for BAUD, CTRL registers from the write operation in Lpuart_Lin_Ip_GotoIdleState.
- **LIN_EXCLUSIVE_AREA_10** is used in function Lin_43_LPUART_FLEXIO_Init to protect the S↩ TAT register from the write operation in Lpuart_Lin_Ip_Init.
- **LIN_EXCLUSIVE_AREA_10** is used in function Lin_43_LPUART_FLEXIO_SendFrame to protect the for STAT register from the write operation in Lpuart_Lin_Ip_AbortTransferData.
- **LIN_EXCLUSIVE_AREA_10** is used in function Lin_43_LPUART_FLEXIO_GoToSleep to protect the for STAT register from the write operation in Lpuart_Lin_Ip_AbortTransferData.
- **LIN_EXCLUSIVE_AREA_10** is used in function Lin_43_LPUART_FLEXIO_GoToSleepInternal to protect the for STAT register from the write operation in Lpuart_Lin_Ip_AbortTransferData.
- **LIN_EXCLUSIVE_AREA_10** is used in function Lin_43_LPUART_FLEXIO_Ipw_Callback to protect the for STAT register from the write operation in Lpuart_Lin_Ip_AbortTransferData.
- **LIN_EXCLUSIVE_AREA_13** is used in function Lin_43_LPUART_FLEXIO_Init to protect the S↩ HIFTCTL, TIMCTL, SHIFTCFG, SHIFTCTL, TIMCFG registers from the write operation in Flexio_Lin↩ _Ip_Init.
- **LIN_EXCLUSIVE_AREA_11** is used in function Lin_43_LPUART_FLEXIO_GoToSleep to protect the for TIMCMP register from the write operation in Flexio_Lin_Ip_GotoIdleState.
- **LIN_EXCLUSIVE_AREA_11** is used in function Lin_43_LPUART_FLEXIO_GoToSleepInternal to protect the for TIMCMP register from the write operation in Flexio_Lin_Ip_GotoIdleState.
- **LIN_EXCLUSIVE_AREA_11** is used in function Lin_43_LPUART_FLEXIO_SendFrame to protect the for TIMCMP register from the write operation in Flexio_Lin_Ip_GotoIdleState.
- **LIN_EXCLUSIVE_AREA_15** is used in function Lin_43_LPUART_FLEXIO_SendFrame to protect the for TIMCTL register from the write operation in Flexio_Lin_Ip_GotoIdleState.
- **LIN_EXCLUSIVE_AREA_15** is used in function Lin_43_LPUART_FLEXIO_GoToSleep to protect the for TIMCTL register from the write operation in Flexio_Lin_Ip_GotoIdleState.
- **LIN_EXCLUSIVE_AREA_15** is used in function Lin_43_LPUART_FLEXIO_GoToSleepInternal to protect the for TIMCTL register from the write operation in Flexio_Lin_Ip_GotoIdleState.

- **LIN_EXCLUSIVE_AREA_16** is used in function Lin_43_LPUART_FLEXIO_SendFrame to protect the TIMCMP, TIMCTL registers from the write operation in Flexio_Lin_Ip_SendFrame.
- **LIN_EXCLUSIVE_AREA_17** is used in function Lin_43_LPUART_FLEXIO_Init to protect the for TIMCTL register from the write operation in Flexio_Lin_Ip_GoToSleepMode.
- **LIN_EXCLUSIVE_AREA_17** is used in function Lin_43_LPUART_FLEXIO_GoToSleepInternal to protect the for TIMCTL register from the write operation in Flexio_Lin_Ip_GoToSleepMode.
- **LIN_EXCLUSIVE_AREA_17** is used in function Flexio_Lin_Ipw_Callback to protect the for TIMCTL register from the write operation in Flexio_Lin_Ip_GoToSleepMode.
- **LIN_EXCLUSIVE_AREA_18** is used in function Lin_43_LPUART_FLEXIO_Init to protect the for TIMCTL, TIMCMP registers from the write operation in Flexio_Lin_Ip_GoToSleepMode.
- **LIN_EXCLUSIVE_AREA_18** is used in function Lin_43_LPUART_FLEXIO_GoToSleepInternal to protect the for TIMCTL, TIMCMP registers from the write operation in Flexio_Lin_Ip_GoToSleepMode.
- **LIN_EXCLUSIVE_AREA_18** is used in function Flexio_Lin_Ipw_Callback to protect the for TIMCTL, TIMCMP registers from the write operation in Flexio_Lin_Ip_GoToSleepMode.
- **LIN_EXCLUSIVE_AREA_20** is used in function Lin_43_LPUART_FLEXIO_Init to protect the C↩ TRL register from the write operation in Lpuart_Lin_Ip_Init.
- **LIN_EXCLUSIVE_AREA_21** is used in function Lin_43_LPUART_FLEXIO_Init to protect the C↩ TRL register from the write operation in Lpuart_Lin_Ip_Init.
- **LIN_EXCLUSIVE_AREA_22** is used in function Lin_43_LPUART_FLEXIO_SetClockMode to protect the updates for BAUD register.
- **LIN_EXCLUSIVE_AREA_23** is used in function Lin_43_LPUART_FLEXIO_Init to protect the B↩ AUD register from the write operation in Lpuart_Lin_Ip_Init.
- **LIN_EXCLUSIVE_AREA_24** is used in function Lin_43_LPUART_FLEXIO_Init to protect the for STAT register from the write operation in Lpuart_Lin_Ip_GoToSleepMode.
- **LIN_EXCLUSIVE_AREA_24** is used in function Lin_43_LPUART_FLEXIO_GoToSleepInternal to protect the for STAT register from the write operation in Lpuart_Lin_Ip_GoToSleepMode.
- **LIN_EXCLUSIVE_AREA_24** is used in function Lin_43_LPUART_FLEXIO_Ipw_Callback to protect the for STAT register from the write operation in Lpuart_Lin_Ip_GoToSleepMode.
- **LIN_EXCLUSIVE_AREA_24** is used in function Lin_43_LPUART_FLEXIO_WakeUpInternal to protect the for STAT register from the write operation in Lpuart_Lin_Ip_GotoIdleState.
- **LIN_EXCLUSIVE_AREA_24** is used in function Lin_43_LPUART_FLEXIO_SendFrame to protect the for STAT register from the write operation in Lpuart_Lin_Ip_GotoIdleState.
- **LIN_EXCLUSIVE_AREA_25** is used in function Lin_43_LPUART_FLEXIO_Init to protect the S↩ TAT register from the write operation in Lpuart_Lin_Ip_Init.
- **LIN_EXCLUSIVE_AREA_25** is used in function Lin_43_LPUART_FLEXIO_SendFrame to protect the STAT register from the write operation in Lpuart_Lin_Ip_SendFrame.
- **LIN_EXCLUSIVE_AREA_25** is used in function Lin_43_LPUART_FLEXIO_Init to protect the for STAT register from the write operation in Lpuart_Lin_Ip_GoToSleepMode.
- **LIN_EXCLUSIVE_AREA_25** is used in function Lin_43_LPUART_FLEXIO_GoToSleepInternal to protect the for STAT register from the write operation in Lpuart_Lin_Ip_GoToSleepMode.
- **LIN_EXCLUSIVE_AREA_25** is used in function Lin_43_LPUART_FLEXIO_Ipw_Callback to protect the for STAT register from the write operation in Lpuart_Lin_Ip_GoToSleepMode.

- **LIN_EXCLUSIVE_AREA_25** is used in function Lin_43_LPUART_FLEXIO_SendFrame to protect the for STAT register from the write operation in Lpuart_Lin_Ip_GotoIdleState.
- **LIN_EXCLUSIVE_AREA_25** is used in function Lin_43_LPUART_FLEXIO_WakeUpInternal to protect the for STAT register from the write operation in Lpuart_Lin_Ip_GotoIdleState.
- **LIN_EXCLUSIVE_AREA_26** is used in function Lin_43_LPUART_FLEXIO_Init to protect the B↩ AUD register from the write operation in Lpuart_Lin_Ip_Init.
- **LIN_EXCLUSIVE_AREA_27** is used in function Lin_43_LPUART_FLEXIO_Init to protect the B↩ AUD, CTRL registers from the write operation in Lpuart_Lin_Ip_Init.

Exclusive Areas implemented in Low level driver layer (IPL)

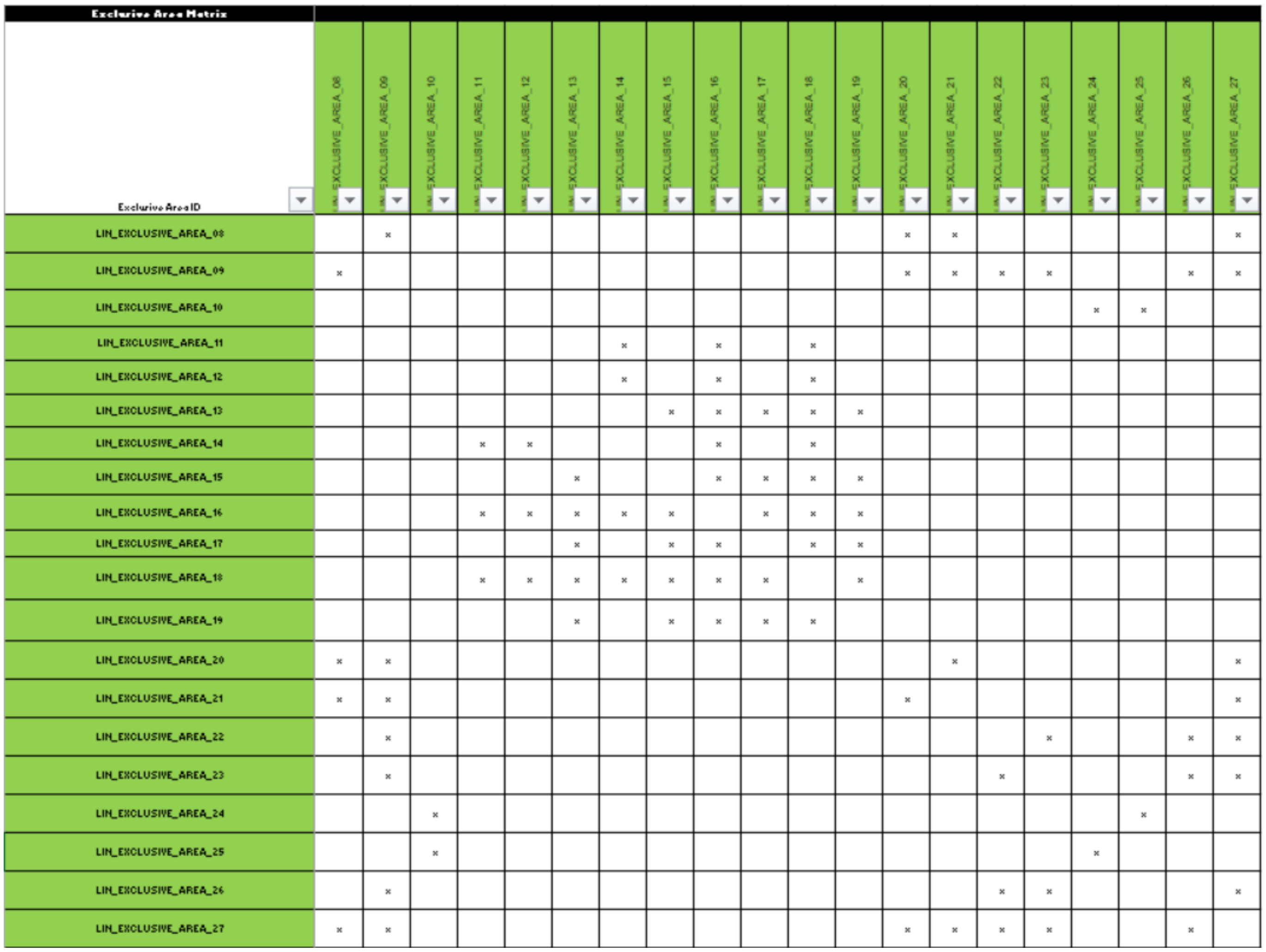
- **LIN_EXCLUSIVE_AREA_08** is used in function Lpuart_Lin_Ip_Init to protect the updates for CTRL register.

- **LIN_EXCLUSIVE_AREA_09** is used in function Lpuart_Lin_Ip_SendFrame to protect the updates for BAUD register.
- **LIN_EXCLUSIVE_AREA_09** is used in function Lpuart_Lin_Ip_Init to protect the updates for BAUD, CTRL registers.
- **LIN_EXCLUSIVE_AREA_09** is used in function Lpuart_Lin_Ip_Deinit to protect the updates for BAUD, CTRL registers.
- **LIN_EXCLUSIVE_AREA_09** is used in function Lpuart_Lin_Ip_AbortTransferData to protect the updates for BAUD, CTRL registers.
- **LIN_EXCLUSIVE_AREA_09** is used in function Lpuart_Lin_Ip_GoToSleepMode to protect the updates for BAUD, CTRL registers.
- **LIN_EXCLUSIVE_AREA_09** is used in function Lpuart_Lin_Ip_GotoIdleState to protect the updates for BAUD, CTRL registers.
- **LIN_EXCLUSIVE_AREA_09** is used in function Lpuart_Lin_Ip_AutoBaudCapture to protect the updates for CTRL register.
- **LIN_EXCLUSIVE_AREA_10** is used in function Lpuart_Lin_Ip_Init to protect the updates for STAT register.
- **LIN_EXCLUSIVE_AREA_10** is used in function Lpuart_Lin_Ip_AbortTransferData to protect the updates for STAT register.
- **LIN_EXCLUSIVE_AREA_13** is used in function Flexio_Lin_Ip_Init to protect the updates for SHI↩ FTCTL, TIMCTL, SHIFTCFG, SHIFTCTL, TIMCFG registers.
- **LIN_EXCLUSIVE_AREA_11** is used in function Flexio_Lin_Ip_GotoIdleState to protect the updates for TIMCMP register.
- **LIN_EXCLUSIVE_AREA_15** is used in function Flexio_Lin_Ip_GotoIdleState to protect the updates for TIMCTL register.
- **LIN_EXCLUSIVE_AREA_16** is used in function Flexio_Lin_Ip_SendFrame to protect the updates for TIMCMP, TIMCTL registers.

- **LIN_EXCLUSIVE_AREA_17** is used in function Flexio_Lin_Ip_GoToSleepMode to protect the updates for TIMCTL register.
- **LIN_EXCLUSIVE_AREA_18** is used in function Flexio_Lin_Ip_GoToSleepMode to protect the updates for TIMCTL, TIMCMP registers.
- **LIN_EXCLUSIVE_AREA_19** is used in function Flexio_Lin_Ip_Deinit to protect the updates for S↩ HIFTCTL, TIMCTL registers.
- **LIN_EXCLUSIVE_AREA_20** is used in function Lpuart_Lin_Ip_Init to protect the updates for CTRL register.
- **LIN_EXCLUSIVE_AREA_20** is used in function Lpuart_Lin_Ip_Deinit to protect the updates for CTRL register.
- **LIN_EXCLUSIVE_AREA_21** is used in function Lpuart_Lin_Ip_Init to protect the updates for CTRL register.
- **LIN_EXCLUSIVE_AREA_21** is used in function Lpuart_Lin_Ip_Deinit to protect the updates for CTRL register.
- **LIN_EXCLUSIVE_AREA_22** is used in function Lpuart_Lin_Ip_Init to protect the updates for BAUD register.
- **LIN_EXCLUSIVE_AREA_22** is used in function Lpuart_Lin_Ip_AutoBaudCapture to protect the updates for BAUD register.
- **LIN_EXCLUSIVE_AREA_23** is used in function Lpuart_Lin_Ip_Init to protect the updates for BAUD register.
- **LIN_EXCLUSIVE_AREA_24** is used in function Lpuart_Lin_Ip_GoToSleepMode to protect the updates for STAT register.
- **LIN_EXCLUSIVE_AREA_24** is used in function Lpuart_Lin_Ip_GotoIdleState to protect the updates for STAT register.
- **LIN_EXCLUSIVE_AREA_25** is used in function Lpuart_Lin_Ip_SendFrame to protect the updates for STAT register.
- **LIN_EXCLUSIVE_AREA_25** is used in function Lpuart_Lin_Ip_AutoBaudCapture to protect the updates for STAT register.
- **LIN_EXCLUSIVE_AREA_25** is used in function Lpuart_Lin_Ip_Init to protect the updates for STAT register.
- **LIN_EXCLUSIVE_AREA_25** is used in function Lpuart_Lin_Ip_GoToSleepMode to protect the updates for STAT register.
- **LIN_EXCLUSIVE_AREA_25** is used in function Lpuart_Lin_Ip_GotoIdleState to protect the updates for STAT register.
- **LIN_EXCLUSIVE_AREA_26** is used in function Lpuart_Lin_Ip_Init to protect the updates for BAUD register.
- **LIN_EXCLUSIVE_AREA_27** is used in function Lpuart_Lin_Ip_Init to protect the updates for BAUD, CTRL registers.

Exclusive Areas are used in Interrupt service request (ISR)

- **LIN_EXCLUSIVE_AREA_09** is used in function ISR(Lpuart_Lin_Ip_IRQHandler) to protect the updates for BAUD, CTRL registers.
- **LIN_EXCLUSIVE_AREA_10** is used in function ISR(Lpuart_Lin_Ip_IRQHandler) to protect the updates for STAT register.
- **LIN_EXCLUSIVE_AREA_11** is used in function ISR(Flexio_Lin_Ip_IrqHandler) to protect the updates for TIMCMP register.
- **LIN_EXCLUSIVE_AREA_15** is used in function ISR(Flexio_Lin_Ip_IrqHandler) to protect the updates for TIMCTL register.
- **LIN_EXCLUSIVE_AREA_24** is used in function ISR(Lpuart_Lin_Ip_IRQHandler) to protect the updates for STAT register.
- **LIN_EXCLUSIVE_AREA_25** is used in function ISR(Lpuart_Lin_Ip_IRQHandler) to protect the updates for STAT register.



Exclusive Area Matrix

Exclusive Area ID	LIN_EXCLUSIVE_AREA_08	LIN_EXCLUSIVE_AREA_09	LIN_EXCLUSIVE_AREA_10	LIN_EXCLUSIVE_AREA_11	LIN_EXCLUSIVE_AREA_12	LIN_EXCLUSIVE_AREA_13	LIN_EXCLUSIVE_AREA_14	LIN_EXCLUSIVE_AREA_15	LIN_EXCLUSIVE_AREA_16	LIN_EXCLUSIVE_AREA_17	LIN_EXCLUSIVE_AREA_18	LIN_EXCLUSIVE_AREA_19	LIN_EXCLUSIVE_AREA_20	LIN_EXCLUSIVE_AREA_21	LIN_EXCLUSIVE_AREA_22	LIN_EXCLUSIVE_AREA_23	LIN_EXCLUSIVE_AREA_24	LIN_EXCLUSIVE_AREA_25	LIN_EXCLUSIVE_AREA_26	LIN_EXCLUSIVE_AREA_27
LIN_EXCLUSIVE_AREA_08		x											x	x						x
LIN_EXCLUSIVE_AREA_09	x												x	x	x	x			x	x
LIN_EXCLUSIVE_AREA_10																	x	x		
LIN_EXCLUSIVE_AREA_11							x		x		x									
LIN_EXCLUSIVE_AREA_12							x		x		x									
LIN_EXCLUSIVE_AREA_13								x	x	x	x	x								
LIN_EXCLUSIVE_AREA_14				x	x				x		x									
LIN_EXCLUSIVE_AREA_15						x			x	x	x	x								
LIN_EXCLUSIVE_AREA_16				x	x	x	x	x		x	x	x								
LIN_EXCLUSIVE_AREA_17						x		x	x		x	x								
LIN_EXCLUSIVE_AREA_18				x	x	x	x	x	x	x		x								
LIN_EXCLUSIVE_AREA_19						x		x	x	x	x									
LIN_EXCLUSIVE_AREA_20	x	x												x						x
LIN_EXCLUSIVE_AREA_21	x	x											x							x
LIN_EXCLUSIVE_AREA_22		x														x			x	x
LIN_EXCLUSIVE_AREA_23		x													x				x	x
LIN_EXCLUSIVE_AREA_24			x															x		
LIN_EXCLUSIVE_AREA_25			x														x			
LIN_EXCLUSIVE_AREA_26		x													x	x				x
LIN_EXCLUSIVE_AREA_27	x	x											x	x	x	x			x	

(Extracted table from RTD_LIN_EXCLUSIVE_AREAS.xlsx)

5.2 Exclusive areas not available on this platform

List of exclusive areas which are not available on this platform (or blank if they're all available).

- **LIN_EXCLUSIVE_AREA_00**
- **LIN_EXCLUSIVE_AREA_01**
- **LIN_EXCLUSIVE_AREA_02**
- **LIN_EXCLUSIVE_AREA_03**
- **LIN_EXCLUSIVE_AREA_04**
- **LIN_EXCLUSIVE_AREA_05**
- **LIN_EXCLUSIVE_AREA_06**
- **LIN_EXCLUSIVE_AREA_07**

5.3 Peripheral Hardware Requirements

The LIN physical interface should be connected to the LIN module pins in order to get the LIN bus voltage levels.

5.4 ISR to configure within AutosarOS - dependencies

ISR Name	HW INT Vector	Observations
LPUART0_LIN_IP_RxTx_IRQHandler	141	None
LPUART1_LIN_IP_RxTx_IRQHandler	142	None
LPUART2_LIN_IP_RxTx_IRQHandler	143	None
LPUART3_LIN_IP_RxTx_IRQHandler	144	None
LPUART4_LIN_IP_RxTx_IRQHandler	145	None
LPUART5_LIN_IP_RxTx_IRQHandler	146	None
LPUART6_LIN_IP_RxTx_IRQHandler	147	None
LPUART7_LIN_IP_RxTx_IRQHandler	148	None
LPUART8_LIN_IP_RxTx_IRQHandler	149	None
LPUART9_LIN_IP_RxTx_IRQHandler	150	None
LPUART10_LIN_IP_RxTx_IRQHandler	151	None
LPUART11_LIN_IP_RxTx_IRQHandler	152	None
LPUART12_LIN_IP_RxTx_IRQHandler	153	None
LPUART13_LIN_IP_RxTx_IRQHandler	154	None
LPUART14_LIN_IP_RxTx_IRQHandler	155	None
LPUART15_LIN_IP_RxTx_IRQHandler	156	None
MCL_FLEXIO_ISR	139	None

Note: LPUARTMSC_LIN_IP_RxTx_IRQHandler will be used for S32K396 device instead of LPUART4_LIN↩_IP_RxTx_IRQHandler.

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 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

Base: The BASE module contains the common files/definitions needed by all RTD modules.

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 LIN driver. This module is required for setting the "LIN Clock Reference" value.

Port: The PORT module is used to configure the port pins with the needed modes, before they are used by the LIN module. For each LINFlex, the TXD and RXD signals need to be configured.

Det: The DET module is used for enabling Development error detection. The API function used is Det_Report↩Error(). The activation/deactivation of Development error detection is configurable using the 'LinDevErrorDetect' configuration parameter.∗

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

EcuM: This module is used for processing the Wakeup notifications of LIN. Whenever the module is in 'Sleep' mode and a wakeup event occurs on a wakeup capable channel, it is reported to EcuM through the EcuM_CheckWakeup↩ Event() API. This is configurable using the 'LinChannelWakeupInfo' configuration parameter.

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 LIN module.

Platform: The PLATFORM module is needed for managing interrupt, system.

EcuC: The ECUC module is used to manage partition and core in multi-core.

OS: The OS module is used to map each partition to a core in multi-core.

LinIf: The LINIF module is used to manage Lin bus.

Mcl: The Mcl module is used to enable the FLEXIO module, to access the common registers and to route the interrupts to the Lin 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

No special measures need to be taken to run LIN module from user mode. The LIN driver code can be executed at any time from both supervisor and user mode.

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 unmarshalling 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.

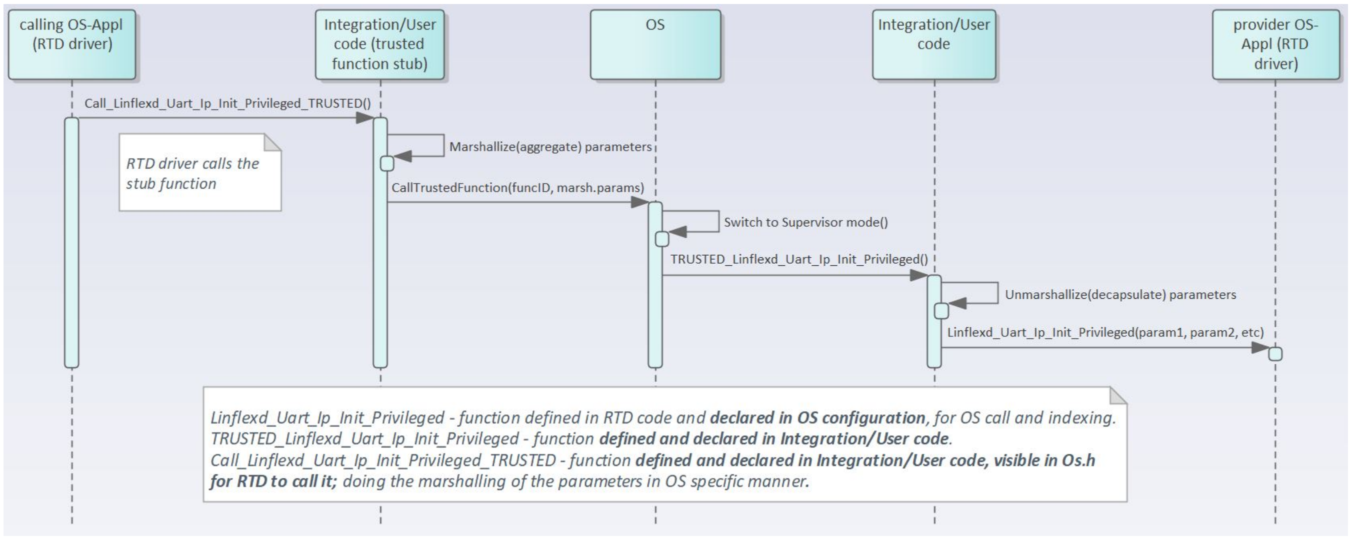

calling OS-Appl (RTD driver)
Integration/User code (trusted function stub)
OS
Integration/User code
provider OS-Appl (RTD driver)
Call_Linflexd_Uart_Ip_Init_Privileged_TRUSTED()
RTD driver calls the stub function
Marshallize(aggregate) parameters
CallTrustedFunction(funcID, marsh.params)
Switch to Supervisor mode()
TRUSTED_Linflexd_Uart_Ip_Init_Privileged()
Unmarshallize(decapsulate) parameters
Linflexd_Uart_Ip_Init_Privileged(param1, param2, etc)
Linflexd_Uart_Ip_Init_Privileged - function defined in RTD code and declared in OS configuration, for OS call and indexing.
TRUSTED_Linflexd_Uart_Ip_Init_Privileged - function defined and declared in Integration/User code.
Call_Linflexd_Uart_Ip_Init_Privileged_TRUSTED - function defined and declared in Integration/User code, visible in Os.h for RTD to call it; doing the marshalling of the parameters in OS specific manner.

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

5.9 Multicore support

1. The LIN implements the "Autosar 4.4 MCAL Multicore Distribution" according to type II, in which the mappable element is set to Channel. For additional details, please refer to AUTOSAR_EXP_BSWDistribution↩ Guide.
2. The Lin and the mappable elements can be allocated to zero, one or several ECUC partitions, by means of "LinEcucPartionRef". If the Lin is mapped to zero ECUC partitions, the Lin behavior reverts to single-core implementation, similar to previous Autosar versions. If the Lin is mapped to one or more ECUC partitions, the Lin enforces the following multi-core assumptions:
   - The LIN 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.
   - The LIN 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 LIN internal variables and the EcucCoreIDs are used for indexing those variables. (AR-86601 Zero based and dense IDs for OS-Cores and OS-Applications).
   - The LIN assumes that initialization is performed on each core, Lin_Init() is called separately for each core, using a different configuration structure.
   - The LIN 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; Lin_Init function will be called with Lin_Config_EcucPartition_1 configuration structure, on Core 1.
   - The LIN 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.
   - The LIN requires that all variables in NonCacheable MemMap sections be allocated accordingly, to avoid data corruption in multicore context.
   - The LIN assumes that RTE module implements the EXCLUSIVE AREAS to be core-aware 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 Lin usecase, to be detailed/removed if some modules require such kind of functionality for critical features which cannot be atomically shared among cores).
   - The LIN assumes that each interrupt is routed by the system only to the core on which is supposed to be serviced.

Chapter 6

Main API Requirements

6.1 Main function calls within BSW scheduler

None.

6.2 API Requirements

None.

6.3 Calls to Notification Functions, Callbacks, Callouts

6.3.1 Call-back Notifications

The LIN Driver uses some callback functions which have to be provided by the respective module:

- EcuM_SetWakeupEvent and EcuM_CheckWakeup have to be provided by the EcuM module.
- LinIf_WakeupConfirmation, LinIf_TxConfirmation, LinIf_RxIndication, LinIf_LinErrorIndication and Lin↩ If_HeaderIndication have to be provided by the LinIf module.

6.3.2 User Notification

None

Chapter 7

Memory allocation

7.1 Sections to be defined in Lin_MemMap.h

Section name	Type of section	Description
LIN_START_SEC_CONFIG_DATA_↩ UNSPECIFIED	Configuration Data	Start of Memory Section for Data Configuration
LIN_STOP_SEC_CONFIG_DATA_U↩ NSPECIFIED	Configuration Data	End of Memory Section for Data Configuration
LIN_START_SEC_CODE	Code	Start of memory Section for Code
LIN_STOP_SEC_CODE	Code	Stop of memory Section for Code
LIN_START_SEC_VAR_CLEARED_8	Variables	Used for variables which have to be aligned to 8 bits. For instance used for variables of size 8 bits or used for composite data types: arrays, structs containing elements of maximum 8 bits. These variables are cleared to zero by start-up code.
LIN_STOP_SEC_VAR_CLEARED_8	Variables	End of the above section.
LIN_START_SEC_VAR_CLEARED_32	Variables	Used for variables which have to be aligned to 8 bits. For instance used for variables of size 8 bits or used for composite data types: arrays, structs containing elements of maximum 32 bits. These variables are cleared to zero by start-up code.
LIN_STOP_SEC_VAR_CLEARED_32	Variables	End of the above section.
LIN_START_SEC_VAR_CLEARED_↩ BOOLEAN	Variables	Used for variables which have to be aligned to 8 bits. For instance used for variables of size 8 bits or used for composite data types: arrays, structs containing elements of boolean. These variables are cleared to zero by start-up code.
LIN_STOP_SEC_VAR_CLEARED_B↩ OOLEAN	Variables	End of the above section.

Section name	Type of section	Description
LIN_START_SEC_VAR_CLEARED_↩ UNSPECIFIED	Variables	Used for variables, structures, arrays when the SIZE (alignment) does not fit the criteria of boolean, 8, 16 or 32 bits.These variables are cleared to zero by start-up code.
LIN_STOP_SEC_VAR_CLEARED_U↩ NSPECIFIED	Variables	End of the above section.
LIN_START_SEC_VAR_INIT_8	Variables	Used for variables which have to be aligned to 8 bits. For instance used for variables of size 8 bits or used for composite data types: arrays, structs containing elements of maximum 8 bits. These variables are initialized with values after every reset
LIN_STOP_SEC_VAR_INIT_8	Variables	End of the above section.
LIN_START_SEC_VAR_INIT_UNSP↩ ECIFIED	Variables	Used for variables, structures, arrays when the SIZE (alignment) does not fit the criteria of boolean, 8, 16 or 32 bitsThese variables are initialized with values after every reset.
LIN_STOP_SEC_VAR_INIT_UNSPE↩ CIFIED	Variables	End of the above section.
LIN_START_SEC_CONST_UNSPECI↩ FIED	Constant Data	Used for constants when the SIZE (alignment) does not fit the criteria of 8,16 or 32 bits.
LIN_STOP_SEC_CONST_UNSPECIF↩ IED	Constant Data	End of the 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 LIN driver into the application.

External Assumption Req ID	External Assumption Text
SWS_Lin_00045	One LIN driver provides access to one LIN hardware unit type (simple U↩ ART or dedicated LIN hardware) that may consist of several LIN channels.
SWS_Lin_00225	There must be at least one statically defined configuration set available for the LIN driver. When the EcuM invokes the initialization function, it has to provide a specific pointer to the configuration that it wishes to use.
SWS_Lin_00014	Each LIN PID shall be associated with a checksum model (either 'enhanced' where the PID is included in the checksum, or 'classic' where only the re-sponse data is check-summed) (see Lin_PduType).
SWS_Lin_00015	Each LIN PID shall be associated with a response data length in bytes (see Lin_PduType).
SWS_Lin_00210	For LIN Master nodes, the upper layer of the LIN Driver has to keep the buffer data consistent until return of function call.For LIN Slave nodes, the upper layer of the LIN Driver has to keep the buffer data consistent until end of response transmission.
SWS_Lin_00245	The content of Lin_GeneralTypes.h shall be protected by a LIN_GENE↩ RAL_TYPES define.
SWS_Lin_00246	If different LIN drivers are used, only one instance of this file has to be included in the source tree. For implementation all Lin_GeneralTypes.h related types in the documents mentioned before shall be considered.
SWS_Lin_00228	Name: - Lin_FramePidType - Type: - uint8 - Range: - 0...0xFE - – - The LIN identifier (0...0x3F) together with its two parity bits. - Description↩ : - Represents all valid protected identifier used by Lin_SendFrame(). - Available via: - Lin_GeneralTypes.h -
SWS_Lin_00229	Name: - Lin_FrameCsModelType - Type: - Enumeration - Range: - LIN↩ _ENHANCED_CS - – - Enhanced checksum model - LIN_CLASSIC_CS - – - Classic checksum model - Description: - This type is used to specify the Checksum model to be used for the LIN Frame. - Available via: - Lin↩ _GeneralTypes.h -

External Assumption Req ID	External Assumption Text
SWS_Lin_00230	Name: - Lin_FrameResponseType - Type: - Enumeration - Range: - L↩ IN_FRAMERESPONSE_TX - – - Response is generated from this node - LIN_FRAMERESPONSE_RX - – - Response is generated from another node and is relevant for this node. - LIN_FRAMERESPONSE_IGNORE - – - Response is generated from another node and is irrelevant for this node - Description: - This type is used to specify whether the frame processor is required to transmit the response part of the LIN frame. - Available via: - Lin_GeneralTypes.h -
SWS_Lin_00231	Name: - Lin_FrameDlType - Type: - uint8 - Range: - 1...8 - – - Data length of a LIN Frame - Description: - This type is used to specify the number of SDU data bytes to copy. - Available via: - Lin_GeneralTypes.h -
SWS_Lin_00232	Name: - Lin_PduType - Type: - Structure - Element: - Lin_FramePidType - Pid - – - Lin_FrameCsModelType - Cs - – - Lin_FrameResponseType - Drc - – - Lin_FrameDlType - Dl - – - uint8∗ - SduPtr - – - Description↩ : - This Type is used to provide PID, checksum model, data length and SDU pointer from the LIN Interface to the LIN driver. - Available via: - Lin_GeneralTypes.h -
SWS_Lin_00233	Name: - Lin_StatusType - Type: - Enumeration - Range: - LIN_NOT_OK - – - LIN frame operation return value. Development or production error occurred - LIN_TX_OK - – - LIN frame operation return value. Successful transmission. - LIN_TX_BUSY - – - LIN frame operation return value. Ongoing transmission (Header or Response). - LIN_TX_HEADER_ER↩ ROR - – - LIN frame operation return value. Erroneous header transmission such as: - Mismatch between sent and read back data - Identifier parity error or - Physical bus error - LIN_TX_ERROR - – - LIN frame operation return value. Erroneous response transmission such as: - Mismatch between sent and read back data - Physical bus error - LIN_RX_OK - – - LIN frame operation return value. Reception of correct response. - LIN_R↩ X_BUSY - – - LIN frame operation return value. Ongoing reception: at least one response byte has been received, but the checksum byte has not been received. - LIN_RX_ERROR - – - LIN frame operation return value. Erroneous response reception such as: - Framing error - Overrun error - Checksum error or - Short response - LIN_RX_NO_RESPONSE - – - L↩ IN frame operation return value. No response byte has been received so far. - LIN_OPERATIONAL - – - LIN channel state return value. Normal operation; the related LIN channel is ready to transmit next header. No data from previous frame available (e.g. after initialization) - LIN_CH_↩ SLEEP - – - LIN channel state return value. Sleep state operation; in this state wake-up detection from slave nodes is enabled. - Description: - LIN operation states for a LIN channel or frame, as returned by the API service Lin_GetStatus(). - Available via: - Lin_GeneralTypes.h -
SWS_Lin_00106	The Lin module's environment shall not call any function of the Lin module before having called Lin_Init except Lin_GetVersionInfo.
SWS_Lin_00193	In case of receiving data the LIN Interface has to wait for the corresponding response part of the LIN frame by polling with the function Lin_GetStatus() after using the function Lin_SendFrame().
SWS_Lin_00194	The Lin module's environment shall only call Lin_SendFrame on a channel which is in state LIN_CH_OPERATIONAL or in one of the sub-states ofLIN_CH_OPERATIONAL.

External Assumption Req ID	External Assumption Text
SWS_Lin_00239	In case of errors during header transmission, it is up to the implementer how to handle these errors (stop/continue transmission) and to decide if the corresponding response is valid or not.
SWS_Lin_00200	The return states LIN_TX_OK, LIN_TX_BUSY, LIN_TX_HEADER↩ _ERROR, LIN_TX_ERROR, LIN_RX_OK, LIN_RX_BUSY, LIN_R↩ X_ERROR , LIN_RX_NO_RESPONSE and LIN_OPERATIONAL are sub-states of the channel state LIN_CH_OPERATIONAL.
SWS_Lin_CONSTR_00279	DRAFT: LinChannel and LinTrcvChannel of one communication channel shall all reference the same ECUC partition.
SWS_Lin_91140	Name: - Lin_SlaveErrorType - Type: - Enumeration - Range: - LIN_E↩ RR_HEADER - – - Error in header - LIN_ERR_RESP_STOPBIT - – - Framing error in response - LIN_ERR_RESP_CHKSUM - – - Checksum error - LIN_ERR_RESP_DATABIT - – - Monitoring error of transmitted data bit in response - LIN_ERR_NO_RESP - – - No response - LIN↩ _ERR_INC_RESP - – - Incomplete response - Description: - This type represents the slave error types that are detected during header reception and response transmission / reception. - Available via: - Lin_General↩ Types.h -
EA_RTD_00042	The application shall not preempt a LIN function working on a channel by calling the same or another LIN function targeting the same channel. Note: The following functions are targeted by the requirement:

- Lin_43_LPUART_FLEXIO_CheckWakeup()
- Lin_43_LPUART_FLEXIO_GoToSleep()
- Lin_43_LPUART_FLEXIO_GoToSleepInternal()
- Lin_43_LPUART_FLEXIO_Wakeup()
- Lin_43_LPUART_FLEXIO_GetStatus()
- Lin_43_LPUART_FLEXIO_SendFrame()

EA_RTD_00043 | The application shall call the function Lin_Init only once during runtime, as stated in SWS↩ _Lin_00146. EA_RTD_00044 | Lin_Init() function shall be called first to initialize the driver. Application shall not call any function of LIN API before the function Lin_Init() has completed. Only Lin_GetVersionInfo() can be called before Lin_Init(). EA_RTD_00045 | The application shall only call Lin_SendFrame on a channel which is in state LIN_CH_OPERATIONAL or in one of the sub-states of LIN_CH_OPERATIONAL. Note: **Rationale**: If Lin_SendFrame() is called while the LIN channel state is LIN_CH_SLEEP, the module might lose its internal consistency. The same risk exists in case of calling Lin_SendFrame() while module state is LIN_NOT_OK.

Implementation Hint: Before calling Lin_SendFrame() the application shall call Lin_GetStatus() in order to ensure that the module state is compliant.

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 Lin is allocated shall be exclusively mapped to a core. Note: Internally, the Lin 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. EA_RTD_00093 | The application shall define EcucCoreIDs in a compact/consecutive order, starting from zero. EA_RTD_00094 | When multicore support is enabled, the application shall call Lin_Init() for each core, using the dedicated configuration pointer for that core. 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 a build a CDD, therefore the BSWMD will not contain reference to the IP interface