Integration Manual

for S32M27X LINTRCV Driver

Document Number: IM34LINTRCVASRR21-11 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

- Supported Derivatives
- Overview
- About This Manual
- Acronyms and Definitions
- Reference List

This integration manual describes the integration requirements for LIN Transceiver 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:

- 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
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
LINTRCV	Lin Transceiver Driver
SPI	Serial Peripheral Interface
ICU	Interrupt Control Unit
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
VLE	Variable Length Encoding
XML	Extensible Markup Language

2.5 Reference List

#	Title	Version
1	Specification of LIN Transceiver Driver	AUTOSAR Release R21-11
2	S32M27x Reference Manual	S32M27x Reference Manual, Rev.2, Draft A, — 02/2023
3	Datasheet	S32M2xx Data Sheet, Rev. 2 RC — 12/2022

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	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 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 LINTRCV driver for S32M27X 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 LINTRCV Driver Files

- LinTrcv_43_AE_TS_T40D34M30I0R0\src\LinTrcv_43_AE.c

- LinTrcv_43_AE_TS_T40D34M30I0R0\src\LinTrcv_43_AE_Ipw.c

- LinTrcv_43_AE_TS_T40D34M30I0R0\src\AeLinPhy_LinTrcv_Ip.c

- LinTrcv_43_AE_TS_T40D34M30I0R0\include\AeLinPhy_LinTrcv_Ip.h

- LinTrcv_43_AE_TS_T40D34M30I0R0\include\AeLinPhy_LinTrcv_Ip_Types.h

- LinTrcv_43_AE_TS_T40D34M30I0R0\include\LinTrcv_43_AE.h

- LinTrcv_43_AE_TS_T40D34M30I0R0\include\LinTrcv_43_AE_Ipw.h

- LinTrcv_43_AE_TS_T40D34M30I0R0\include\LinTrcv_43_AE_Ipw_Types.h

- LinTrcv_43_AE_TS_T40D34M30I0R0\include\LinTrcv_43_AE_Types.h

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

- LinTrcv_43_AE_Cfg.c

- LinTrcv_43_AE_Cfg.h

- LinTrcv_43_AE_Ipw_Cfg.h

- LinTrcv_43_AE_Ipw_Cfg.c

- AeLinPhy_LinTrcv_Ip_Cfg.h

- AeLinPhy_LinTrcv_Ip_Cfg.c

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

- Ae_Ipw_PBcfg.h
- Ae_PBcfg.h
- Aec_Ip_PBcfg.h
- Ae_Ipw_PBcfg.c
- Ae_PBcfg.c
- Aec_Ip_PBcfg.c
- Ae_Cfg.h
- Ae_Ipw_Cfg.h
- Aec_Ip_Cfg.h
- Aec_Ip_Cfg_Defines.h
- Aec_Ip_Cfg.c

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

- Spi_PBcfg.c
- Spi_Ipw_PBcfg.c
- Lpspi_Ip_PBcfg.c
- Spi_PBcfg.h
- Spi_Ipw_PBcfg.h
- Lpspi_Ip_PBcfg.h
- Lpspi_Ip_Cfg.h
- Spi_Cfg.h
- Spi_Ipw_Cfg.h

3.2.5   BASE Files:

- Base_TS_T40D34M30I0R0\include\Lin_GeneralTypes.h
- Base_TS_T40D34M30I0R0\include\Mcal.h
- Base_TS_T40D34M30I0R0\include\LinTrcv_MemMap.h
- Base_TS_T40D34M30I0R0\include\Platform_Types.h
- Base_TS_T40D34M30I0R0\include\Soc_Ips.h
- Base_TS_T40D34M30I0R0\include\StandardTypes.h
- Base_TS_T40D34M30I0R0\include\OsIf.h
- Base_TS_T40D34M30I0R0\include\Devassert.h
- Base_TS_T40D34M30I0R0\generate_PC\include\modules.h
- Base_TS_T40D34M30I0R0\header\S32M24x_AEC_AE.h

3.2.6 DET Files:

- Det_TS_T40D34M30I0R0\include\Det.h

- Det_TS_T40D34M30I0R0\src\Det.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 AE Driver Files:

- Ae_TS_T40D2M19I0R0\src\Ae.c

- Ae_TS_T40D2M19I0R0\src\Ae_Ipw.c

- Ae_TS_T40D2M19I0R0\src\Aec_Ip.c

- Ae_TS_T40D2M19I0R0\src\Ae_Ip_Hw_Access.c

- Ae_TS_T40D2M19I0R0\include\Hvm_Ip_Hw_Access.c

- Ae_TS_T40D2M19I0R0\include\Hvm_Ip.c

- Ae_TS_T40D2M19I0R0\include\Ae.h

- Ae_TS_T40D2M19I0R0\include\Ae_Ipw.h

- Ae_TS_T40D2M19I0R0\include\Aec_Ip.h

- Ae_TS_T40D2M19I0R0\include\Aec_Ip_Types.h

- Ae_TS_T40D2M19I0R0\include\Aec_Ip_Hw_Access.h

- Ae_TS_T40D2M19I0R0\include\Hvm_Ip_Hw_Access.h

- Ae_TS_T40D2M19I0R0\include\Hvm_Ip.h

- Ae_TS_T40D2M19I0R0\include\Hvm_Ip_Types.h

3.2.10 Spi Files:

- Spi_TS_T40D2M19I0R0\include\Spi.h

- Spi_TS_T40D2M19I0R0\include\Spi_Ipw.h

- Spi_TS_T40D2M19I0R0\src\Spi.c

- Spi_TS_T40D2M19I0R0\src\Spi_Ipw.c

- Spi_TS_T40D2M19I0R0\src\Lpspi_Ip.c

- Spi_TS_T40D2M19I0R0\include\Lpspi_Ip.h

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:
  - LinTrcv_43_AE_TS_T40D34M30I0R0\config\LinTrcv.xdm
- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - LinTrcv_43_AE_TS_T40D34M30I0R0\autosar\LinTrcv_<subderivative_name>.epd
- Code Generation Templates for parameters without variation points:
  - LinTrcv_43_AE_TS_T40D34M30I0R0\generate_PC\include\LinTrcv_Cfg.h
  - LinTrcv_43_AE_TS_T40D34M30I0R0\generate_PC\include\LinTrcv_Ipw_Cfg.h
  - LinTrcv_43_AE_TS_T40D34M30I0R0\generate_PC\include\AeLinPhy_LinTrcv_Ip_Cfg.h
  - LinTrcv_43_AE_TS_T40D34M30I0R0\generate_PC\include\LinTrcv_Cfg.c
  - LinTrcv_43_AE_TS_T40D34M30I0R0\generate_PC\include\LinTrcv_Ipw_Cfg.c
  - LinTrcv_43_AE_TS_T40D34M30I0R0\generate_PC\include\AeLinPhy_LinTrcv_Ip_Cfg.c

3.3.2 Steps to generate the configuration:

1. Copy the following module folders into the Tresos plugins folder:
   - LinTrcv_43_AE_TS_T40D34M30I0R0
   - Base_TS_T40D34M30I0R0
   - Det_TS_T40D34M30I0R0
   - EcuC_TS_T40D34M30I0R0
   - EcuM_TS_T40D34M30I0R0
   - Resource_TS_T40D34M30I0R0
   - Mcu_TS_T40D34M30I0R0
   - LinIf_TS_T40D34M30I0R0
   - Ae_TS_T40D34M30I0R0
   - Spi_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

The LinTrcv Driver shall be initialized through the LinTrcv_43_AE_Init API function. The Mcu, Port, Spi, Ae modules shall be initialized beforehand.

4.2 Function Calls during Shutdown

There is no shutdown specific procedure for Lin Transceiver driver.

4.3 Function Calls during Wake-up

Wakeup notification must be supported by Lin Transceiver driver. There are two wake-up methods which can be notified via LIN Trcv driver: internal wakeup which comes from an upper layer of the LIN Stack or external wakeup which comes from the LIN Bus (signal from other Lin node in the Network).

4.3.1 Internal wakeup

An internal wakeup is initiated by an upper layer, e.g. by calling LinTrcv_SetOpMode.

4.3.2 External wakeup

The wake up can be detected by LIN transceiver using either an interrupt and polling.

Function calls to module

4.3.2.1 Lin Trcv Wakeup sign Interrogation by Polling Method

Lin Trasceiver supports Wakeup by Polling The ECU State Manager module will regularly check the LIN Interface module, which in turn interogates LIN TRCV module for a wakeup signal. The sequence diagram below describes the Wakeup By Polling Flow in the LIN Autosar Stack:
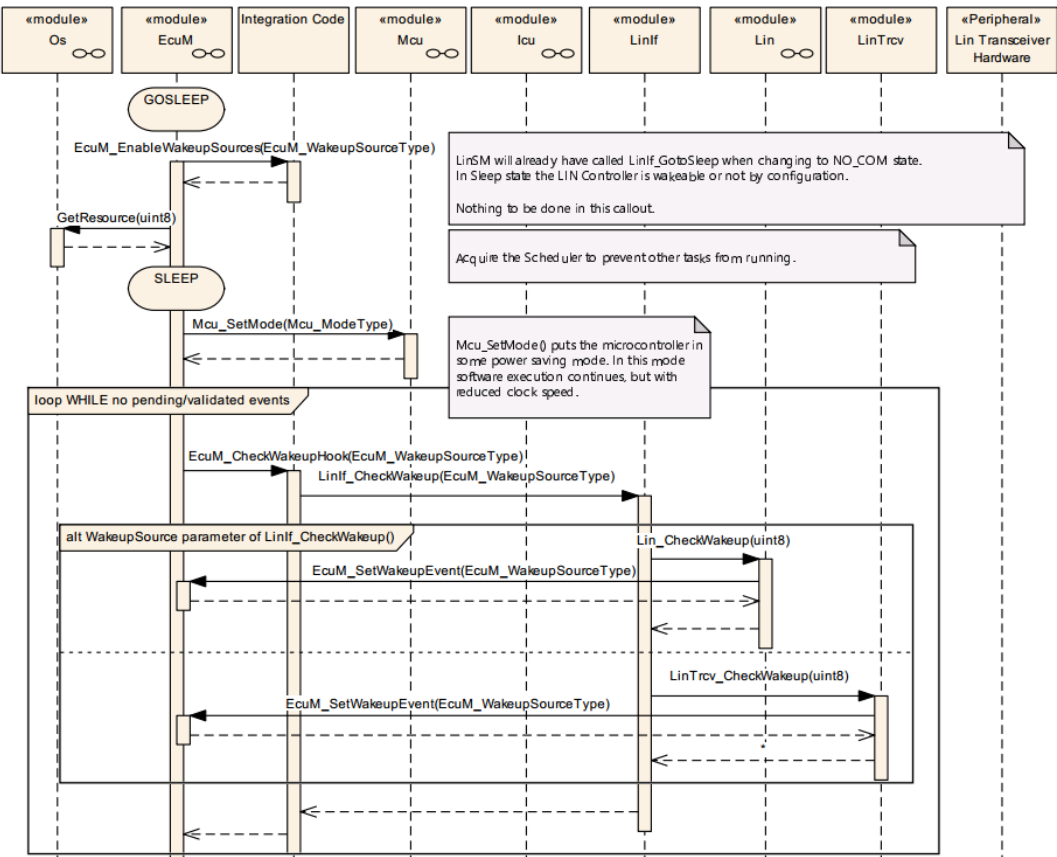
Figure 4.1 Wake up by Polling

4.3.2.2 Lin Trcv Wakeup sign Detection by Interrupt Method

The LIN transceiver can detect wakeup signal from LIN Bus and signal it to upper layers via interrupt. Please refer to the sequence diagram below describes the Wakeup By Interrupt Flow in the LIN Autosar Stack for more detail.
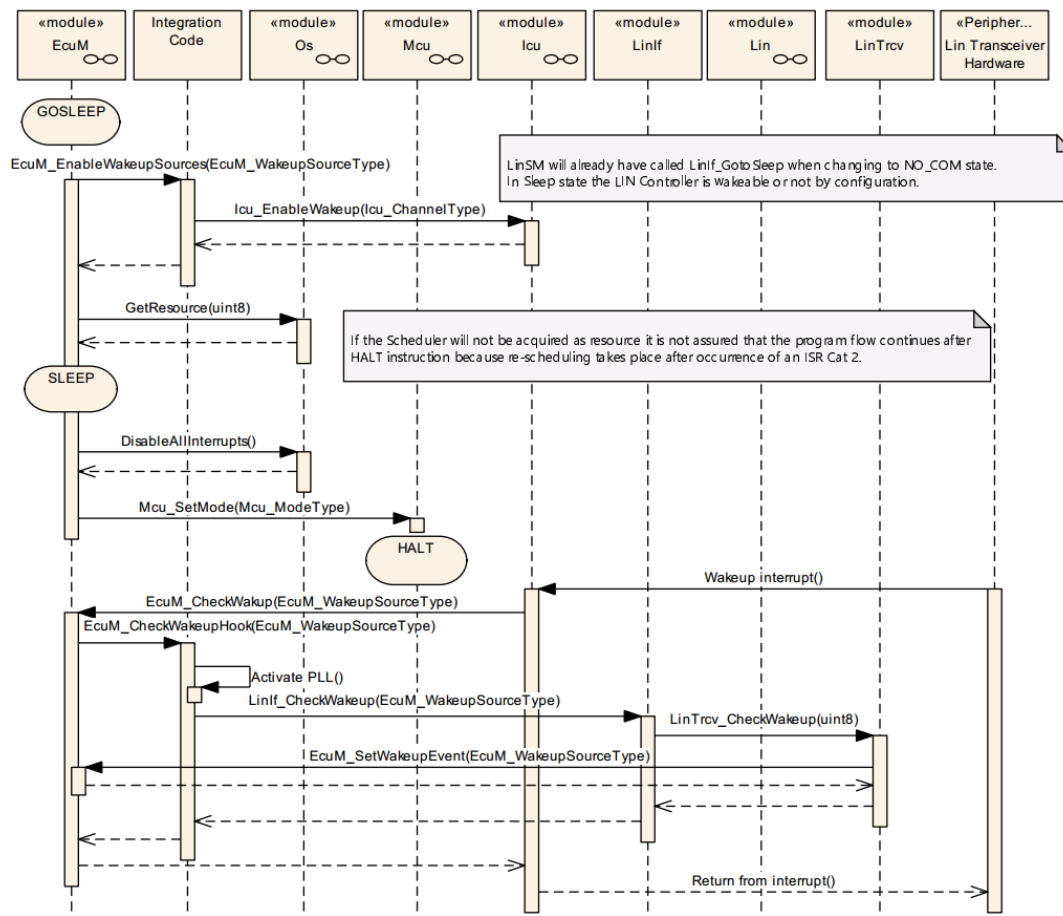
Figure 4.2 Wake up by Interrupt

Chapter 5

Module requirements

5.1   Exclusive areas to be defined in BSW scheduler

No exclusive areas have been identified.

5.2   Exclusive areas not available on this platform

None

5.3   Peripheral Hardware Requirements

None

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 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.

Port: The PORT module is used to configure the port pins with the needed modes, before they are used by the LIN amd LINTRCV modules. The TXD and RXD signals need to be configured in order connect LIN to LINTRCV.

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.∗

EcuM: This module is used for processing the Wakeup notifications of LINTRCV. 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_Check↩ WakeupEvent() API.

Resource: RESOURCE module is used to select microcontroller's derivatives.

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: All LIN transceiver drivers are arranged below LinIf.

Spi: One SPI sequence used for AE module in order access to Lin transceiver register.

Ae: The Ae module is used to access LIN transceiver hardware connected via Spi.

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↩
TRCV module from user mode. The LINTRCV 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 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.
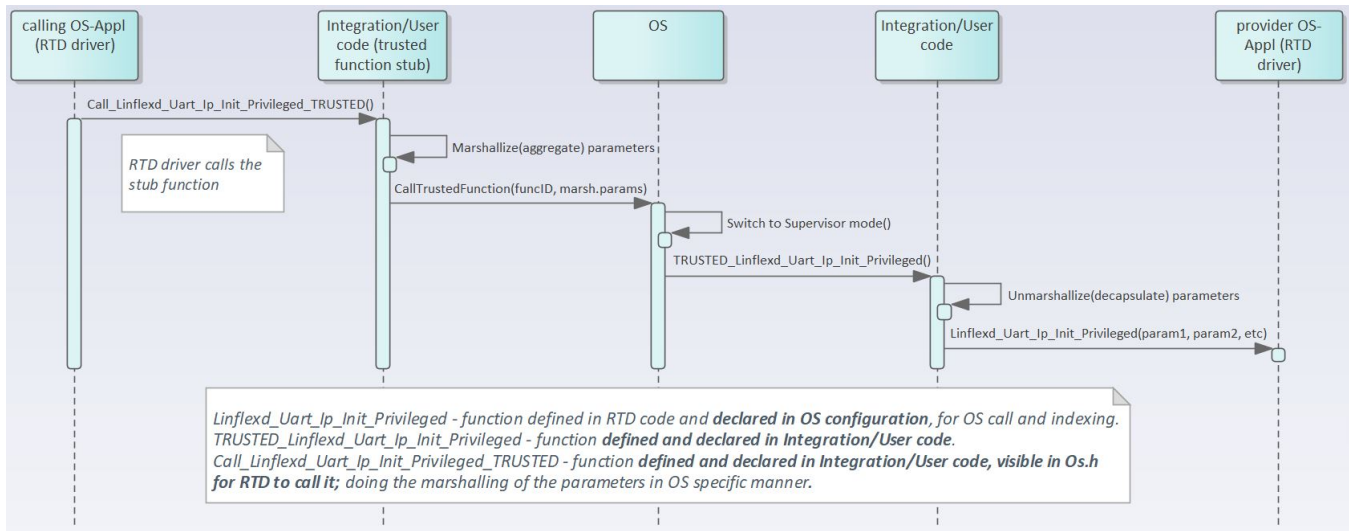

Figure 5.1 Example sequence chart for calling `Linflexd_Uart_Ip_Init_Privileged` as trusted function

5.9 Multicore support

S32M27x family is single core MCU, so this feature is not available on this kind of platform.

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
None

6.3.2   User Notification
None

Chapter 7

Memory allocation

- Sections to be defined in LinTrcv_MemMap.h

- Linker command file

7.1   Sections to be defined in LinTrcv_MemMap.h

Section name	Type of section	Description
LINTRCV_43_AE_START_SEC_CO↩NFIG_DATA_UNSPECIFIED	Configuration Data	Start of Memory Section for Data Configuration
LINTRCV_43_AE_STOP_SEC_CON↩FIG_DATA_UNSPECIFIED	Configuration Data	End of Memory Section for Data Configuration
LINTRCV_43_AE_START_SEC_CO↩DE	Code	Start of memory Section for Code
LINTRCV_43_AE_STOP_SEC_CODE	Code	Stop of memory Section for Code
LINTRCV_43_AE_START_SEC_VA↩R_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.
LINTRCV_43_AE_STOP_SEC_VAR↩_CLEARED_8	Variables	End of the above section.
LINTRCV_43_AE_START_SEC_VA↩R_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.
LINTRCV_43_AE_STOP_SEC_VAR↩_CLEARED_UNSPECIFIED	Variables	End of the above section.
LINTRCV_43_AE_START_SEC_VA↩R_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

Section name	Type of section	Description
LINTRCV_43_AE_STOP_SEC_VAR←_INIT_8	Variables	End of the above section.
LINTRCV_43_AE_START_SEC_CO←NST_UNSPECIFIED	Constant Data	Used for constants when the SIZE (alignment) does not fit the criteria of 8,16 or 32 bits.
LINTRCV_43_AE_STOP_SEC_CON←ST_UNSPECIFIED	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

Note

　　None.