Integration Manual

for S32K14X LIN Driver

Document Number: IM2LINASR4.2 Rev0002R1.0.2
Rev. 1.0

Contents

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

Chapter 11
External Assumptions for LIN driver

Chapter 1
Revision History

Table 1-1. Revision History

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

Chapter 2
Introduction

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

2.1 Supported Derivatives

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

Table 2-1. S32K14X Derivatives

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

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

2.2 Overview

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

AUTOSAR

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

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

2.3 About this Manual

This Technical Reference employs the following typographical conventions:

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

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

Notes and warnings are shown as below:

Note

This is a note.

2.4 Acronyms and Definitions

Table 2-2. Acronyms and Definitions

Term	Definition
API	Application Programming Interface
ASM	Assembler
AUTOSAR	Automotive Open System Architecture
BSMI	Basic Software Make file Interface
C/CPP	C and C++ Source Code
DEM	Diagnostic Event Manager
DET	Default Error Tracer
EcuM	ECU state Manager
GUI	Graphical User Interface
ISR	Interrupt Service Routine
LIN	Local Interconnect Network
MCU	Micro Controller Unit
N/A	Not Applicable

Table continues on the next page...

Table 2-2. Acronyms and Definitions (continued)

Term	Definition
OS	Operating System
PB Variant	Post Build Variant
PC Variant	Pre Compile Variant
VLE	Variable Length Encoding

2.5 Reference List

Table 2-3. Reference List

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

Chapter 3
Building the Driver

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

3.1 Build Options

The LIN driver files are compiled using

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

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

Note

The TS_T40D2M10I2R0 plugin name is composed as follow:

TS_T = Target_Id

D = Derivative_Id

M = SW_Version_Major

I = SW_Version_Minor

R = Revision

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

3.1.1 GHS Compiler/Linker/Assembler Options

Table 3-1. Compiler Options

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

Table 3-2. Assembler Options

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

Table 3-3. Linker Options

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

3.1.2 GCC Compiler/Linker/Assembler Options

Table 3-4. Compiler Options

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

Table continues on the next page...

Table 3-4. Compiler Options (continued)

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

Table 3-5. Assembler Options

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

Table 3-6. Linker Options

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

Table continues on the next page...

Table 3-6. Linker Options (continued)

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

3.1.3 IAR Compiler/Linker/Assembler Options

Table 3-7. Compiler Options

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

Table 3-8. Assembler Options

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

Table 3-9. Linker Options

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

3.2 Files required for Compilation

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

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

LIN Files

- ..\Lin_TS_T40D2M10I2R0\src\Lin.c
- ..\Lin_TS_T40D2M10I2R0\src\Lin_IPW.c
- ..\ Lin_TS_T40D2M10I2R0\src\Lin_LPUART.c
- ..\ Lin_TS_T40D2M10I2R0\src\Lin_LPUART_Irq.c
- ..\ Lin_TS_T40D2M10I2R0\src\Lin_FlexIO.c
- ..\ Lin_TS_T40D2M10I2R0\src\Lin_NonASR.c

- ..\Lin_TS_T40D2M10I2R0\include\Lin.h
- ..\Lin_TS_T40D2M10I2R0\include\Lin_NonASR.h
- ..\Lin_TS_T40D2M10I2R0\include\Lin_IPW.h

- ..\ Lin_TS_T40D2M10I2R0\include\Lin_LPUART.h
- ..\ Lin_TS_T40D2M10I2R0\include\Lin_FlexIO.h
- ..\ Lin_TS_T40D2M10I2R0\include\Reg_eSys_LPUART.h

LIN Generated Files

- Lin_[VariantName]_PBcfg.c - This file should be generated by the user using a configuration tool for compilation. The file contains the definition of the init pointer for the respective variant.
- Lin_Cfg.c - This file should be generated by the user using a configuration tool for compilation.
- Lin_Cfg.h - This file should be generated by the user using a configuration tool for compilation.

Note

As a deviation from standard:

- Lin_[VariantName]_PBcfg.c files will contain the definition for all parameters that are variant aware, independent of the configuration class that will be selected (PC, PB)

- Lin_Cfg.c file will contain the definition for all configuration structures containing only variables that are not variant aware, configured and generated only once. This file alone does not contain the whole structure needed by Lin_Init function to configure the driver. Based on the number of variants configured in the EcuC, there can be more than one configuration structure for one module even for PreCompile variant.

Files from Base common folder

- ..\Base_TS_T40D2M10I2R0\generate_PC\include\modules.h
- ..\Base_TS_T40D2M10I2R0\generate_PC\include\Soc_Ips.h
- ..\Base_TS_T40D2M10I2R0\include\Cer.h
- ..\Base_TS_T40D2M10I2R0\include\Compiler.h
- ..\Base_TS_T40D2M10I2R0\include\Compiler_Cfg.h
- ..\Base_TS_T40D2M10I2R0\include\ComStack_Cfg.h
- ..\Base_TS_T40D2M10I2R0\include\ComStack_Types.h
- ..\Base_TS_T40D2M10I2R0\include\Lin_GeneralTypes.h
- ..\Base_TS_T40D2M10I2R0\include\Mcal.h
- ..\Base_TS_T40D2M10I2R0\include\Lin_MemMap.h
- ..\Base_TS_T40D2M10I2R0\include\Platform_Types.h
- ..\Base_TS_T40D2M10I2R0\include\Reg_eSys.h
- ..\Base_TS_T40D2M10I2R0\include\Reg_Macros.h
- ..\Base_TS_T40D2M10I2R0\include\Std_Types.h

Files from Dem folder:

- ..\Dem_TS_T40D2M10I2R0\generate_PC\include\Dem_IntErrId.h
- ..\Dem_TS_T40D2M10I2R0\include\Dem.h
- ..\Dem_TS_T40D2M10I2R0\include\Dem_Types.h

Files from Det folder:

- ..\Det_TS_T40D2M10I2R0\include\Det.h

Files from EcuM folder:

- ..\EcuM_TS_T40D2M10I2R0\generate_PC\include\EcuM_Cfg.h
- ..\EcuM_TS_T40D2M10I2R0\include\EcuM.h
- ..\EcuM_TS_T40D2M10I2R0\include\EcuM_Cbk.h

Files from Mcl folder:

- ..\Mcl_TS_T40D2M10I2R0\include\Reg_eSys_FlexIO.h
- ..\Mcl_TS_T40D2M10I2R0\src\FlexIO_Common.c

Files from Rte folder:

- ..\Rte_TS_T40D2M10I2R0\include\SchM_Lin.h
- ..\Rte_TS_T40D2M10I2R0\src\SchM_Lin.c

3.3 Setting up the Plug-ins

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

Location of various files inside the LIN module folder:

- VSMD (Vendor Specific Module Definition) file in EB tresos Studio XDM format:
  - ..\Base_TS_T40D2M10I2R0\config\Base.xdm
  - ..\EcuM_TS_T40D2M10I2R0\config\EcuM.xdm
  - ..\EcuC_TS_T40D2M10I2R0\config\EcuC.xdm
  - ..\Lin_TS_T40D2M10I2R0\config\Lin.xdm
  - ..\Mcu_TS_T40D2M10I2R0\config\Mcu.xdm
  - ..\Resource_TS_T40D2M10I2R0\config\Resource.xdm
- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - ..\Base_TS_T40D2M10I2R0\autosar\Base.epd
  - ..\EcuM_TS_T40D2M10I2R0\autosar\EcuM.epd
  - ..\EcuC_TS_T40D2M10I2R0\autosar\EcuC.epd
  - ..\Lin_TS_T40D2M10I2R0\autosar\Lin.epd

    - ..\Mcu_TS_T40D2M10I2R0\autosar\Mcu.epd
    - ..\Resource_TS_T40D2M10I2R0\autosar\Resource.epd

- Code Generation Templates for parameters without variation points:
    - ..\Lin_TS_T40D2M10I2R0\generate_PC\src\Lin_Cfg.c
    - ..\Lin_TS_T40D2M10I2R0\generate_PC\include\Lin_Cfg.h
    - ..\Lin_TS_T40D2M10I2R0\generate_PC\Lin_VersionCheck_Inc.m
    - ..\Lin_TS_T40D2M10I2R0\generate_PC\Lin_VersionCheck_Src.m
    - ..\Lin_TS_T40D2M10I2R0\generate_PC\Lin_BaudRate_Comp.m

- Code Generation Templates for variant aware parameters:
    - ..\Lin_TS_T40D2M10I2R0\generate_PB\src\Lin_PBcfg.c
    - ..\Lin_TS_T40D2M10I2R0\generate_PB\Lin_VersionCheck_Inc.m
    - ..\Lin_TS_T40D2M10I2R0\generate_PB\Lin_VersionCheck_Src_PB.m
    - ..\Lin_TS_T40D2M10I2R0\generate_PB\Lin_BaudRate_Comp.m

Steps to generate the configuration:

1. Copy the module folders Lin_TS_T40D2M10I2R0, Base_TS_T40D2M10I2R0,Resource_TS_T40D2M10I2R0, EcuM_TS_T40D2M10I2R0, EcuC_TS_T40D2M10I2R0,Mcu_TS_T40D2M10I2R0 into the Tresos plugins folder.
2. Set the desired Tresos Output location folder for the generated sources and header files.
3. Use the EB tresos Studio GUI to modify ECU configuration parameters values.
4. Generate the configuration files.

Dependencies

- **MCU** is required to use System Clock when clock source is used as Peripheral clock source to generate LIN Segment values.
- **RESOURCE** is required to select processor derivative. Current Lin driver has support for the following derivatives, everyone having attached a Resource file: s32k148_lqfp144, s32k148_lqfp176, s32k148_mapbga100, s32k146_lqfp144, s32k146_lqfp100, s32k146_lqfp64, s32k146_mapbga100, s32k144_lqfp100, s32k144_lqfp64, s32k144_mapbga100, s32k142_lqfp100, s32k142_lqfp64, s32k118_lqfp48, s32k118_lqfp64, s32k142_lqfp48, s32k144_lqfp48, s32k148_lqfp100 .
- **ECUM** is required for selecting the reference to the wakeup source for every Lin controller.
- **ECUC** is required for selecting the postbuild variant criterion.
- **DET** is required for signaling the default error detection (parameters out of range, null pointers, etc).
- **DEM** is required for signaling the production error detection (hardware failure, etc).

Resource Parameters Configuration

1. **Lin.LinGlobalConfig.LinChannel** - number of maximum available LPUART/ FLEXIO controllers on chip.
2. **Lin.LinGlobalConfig.LinChannel.LinHwChannel** - list of available LPUART/ FLEXIO controllers on chip (LPUART_0,LPUART_1,LPUART_2,LPUART_3,FLEXIO_0_LIN_0,FLEXIO_0_LIN_1).
3. **LinExternalWKPUSupport** - TRUE if wake up support needs to be configured.
4. **LinExternalWKPUChannelID** - External Wake up channel(s) assigned for each LIN HW channel.

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_Init(). The MCU module should be initialized before the LIN is initialized. The Lin driver does not need OS Support except for ISR's. Hence, can be initialized either in STARTUP1 or STARTUP2 phase of EcuM initialization. This depends on the implementation, desired duration for STARTUP1 & Target hardware design. The LIN module shall be initialized by Lin_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 LPUART/ FLEXIO 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 have to be initialized as well before using it. This is also done by the Lin_Init(<&Lin_Configuration>) service.

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

Lin_GoToSleep(LIN_CHANNEL): which put the LIN driver into sleep mode and send a Go-to-sleep command over the bus.

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_Wakeup(LIN_CHANNEL) API service may be used.

External Wakeup:

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

If the channel is configured with "wake-up support", upon wakeup detection on Rx pin of the configured LPUART channel the ISR "Lin_LPUART_TxRxInterruptHandler(*LIN channel*)" will be executed (based on the LIN Channel configured) to wake-up Lin Driver.

If the channel is not configured with "wake-up support", the Lin stack may call Lin_CheckWakeup(LIN_CHANNEL) service API in order to identify if a slave on the Lin bus issued a wake-up request. In case such a request is identified then, it is the Lin stack responsibility to wake up the channel and process the slave request.

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:

LIN_EXCLUSIVE_AREA_00 To protect the LPUART_BAUD, LPUART_STAT, LPUART_CTRL registers during the read/modify/write action. It is used in the function Lin_LPUART_InitChannel.

LIN_EXCLUSIVE_AREA_01 To protect the LPUART_CTRL register during the read/modify/write action. It is used in the function Lin_LPUART_SendHeader.

LIN_EXCLUSIVE_AREA_02 To protect the LPUART_CTRL register during the read/modify/write action. It is used in the function Lin_LPUART_SendHeader.

LIN_EXCLUSIVE_AREA_03 To protect the LPUART_CTRL register during the read/modify/write action. It is used in the function Lin_LPUART_SendResponse.

LIN_EXCLUSIVE_AREA_04 To protect the LPUART_CTRL register during the read/modify/write action. It is used in the function Lin_LPUART_GoToSleep.

LIN_EXCLUSIVE_AREA_05 To protect the LPUART_CTRL register during the read/modify/write action. It is used in the function Lin_LPUART_GoToSleep.

LIN_EXCLUSIVE_AREA_06 To protect the LPUART_BAUD, LPUART_STAT, LPUART_CTRL registers during the read/modify/write action. It is used in the function Lin_LPUART_GoToSleepInternal.

LIN_EXCLUSIVE_AREA_07 To protect the LPUART_BAUD, LPUART_STAT, LPUART_CTRL registers during the read/modify/write action. It is used in the function Lin_LPUART_GoToSleepInternal.

LIN_EXCLUSIVE_AREA_08 To protect the LPUART_BAUD, LPUART_STAT, LPUART_CTRL registers during the read/modify/write action. It is used in the function Lin_LPUART_WakeUp.

LIN_EXCLUSIVE_AREA_09 To protect the LPUART_BAUD, LPUART_STAT, LPUART_CTRL registers during the read/modify/write action. It is used in the function Lin_LPUART_WakeupInternal.

LIN_EXCLUSIVE_AREA_10 To protect the LPUART_CTRL register during the read/modify/write action. It is used in the function Lin_LPUART_TxRxInterruptHandler.

LIN_EXCLUSIVE_AREA_11 To protect the LPUART_STAT, LPUART_CTRL registers during the read/modify/write action. It is used in the function Lin_LPUART_TxRxInterruptHandler.

LIN_EXCLUSIVE_AREA_12 To protect the LPUART_CTRL register during the read/modify/write action. It is used in the function Lin_LPUART_TxRxInterruptHandler.

LIN_EXCLUSIVE_AREA_13 To protect the LPUART_CTRL register during the read/modify/write action. It is used in the function Lin_LPUART_TxRxInterruptHandler.

LIN_EXCLUSIVE_AREA_14 To protect the LPUART_CTRL register during the read/modify/write action. It is used in the function Lin_LPUART_TxRxInterruptHandler.

LIN_EXCLUSIVE_AREA_15 To protect the LPUART_CTRL register during the read/modify/write action. It is used in the function Lin_LPUART_TxRxInterruptHandler.

LIN_EXCLUSIVE_AREA_16 To protect the LPUART_BAUD, LPUART_STAT, LPUART_CTRL registers during the read/modify/write action. It is used in the function Lin_LPUART_TxRxInterruptHandler.

LIN_EXCLUSIVE_AREA_17 To protect the LPUART_CTRL register during the read/modify/write action. It is used in the function Lin_LPUART_TxRxInterruptHandler.

LIN_EXCLUSIVE_AREA_18 To protect the LPUART_CTRL register during the read/modify/write action. It is used in the function Lin_LPUART_TxRxInterruptHandler.

LIN_EXCLUSIVE_AREA_19 To protect the LPUART_BAUD, LPUART_STAT, LPUART_CTRL registers during the read/modify/write action. It is used in the function Lin_LPUART_TxRxInterruptHandler.

LIN_EXCLUSIVE_AREA_20 To protect the LPUART_STAT, LPUART_CTRL registers during the read/modify/write action. It is used in the function Lin_LPUART_TxRxInterruptHandler.

LIN_EXCLUSIVE_AREA_21 To protect the LPUART_STAT registers during the read/modify/write action. It is used in the function Lin_LPUART_TxRxInterruptHandler.

LIN_EXCLUSIVE_AREA_22 To protect the LPUART_STAT, LPUART_CTRL registers during the read/modify/write action. It is used in the function Lin_LPUART_TxRxInterruptHandler.

LIN_EXCLUSIVE_AREA_23 To protect the LPUART_STAT register during the read/modify/write action. It is used in the function Lin_LPUART_ErrorInterruptHandler.

LIN_EXCLUSIVE_AREA_24 To protect the LPUART_STAT register during the read/modify/write action. It is used in the function Lin_LPUART_ErrorInterruptHandler.

LIN_EXCLUSIVE_AREA_25 To protect the LPUART_BAUD, LPUART_CTRL register during the read/modify/write action. It is used in the function Lin_SetClockMode.

Table 5-1. Exclusive Areas

Exclusive Area ID	LIN_EXCLUSIVE_AREA_00	LIN_EXCLUSIVE_AREA_01/ 02	LIN_EXCLUSIVE_AREA_03	LIN_EXCLUSIVE_AREA_04/05	LIN_EXCLUSIVE_AREA_06/07	LIN_EXCLUSIVE_AREA_08	LIN_EXCLUSIVE_AREA_09	LIN_EXCLUSIVE_AREA_10->22	LIN_EXCLUSIVE_AREA_23/24	LIN_EXCLUSIVE_AREA_25
LIN_EXCLUSIVE_AREA_00		x	x	x	x	x	x	x	x	x
LIN_EXCLUSIVE_AREA_01 / 02	x		x	x	x	x	x	x	x	x
LIN_EXCLUSIVE_AREA_03	x	x		x	x	x	x	x	x	x
LIN_EXCLUSIVE_AREA_04 /05	x	x	x		x	x	x	x	x	x
LIN_EXCLUSIVE_AREA_06 / 07	x	x	x	x		x	x	x	x	x
LIN_EXCLUSIVE_AREA_08	x	x	x	x	x		x	x	x	x
LIN_EXCLUSIVE_AREA_09	x	x	x	x	x	x		x	x	x
LIN_EXCLUSIVE_AREA_10 ->22	x	x	x	x	x	x	x		x	x
LIN_EXCLUSIVE_	x	x	x	x	x	x	x	x		x

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

Exclusive Area ID	LIN_EXCLUSIVE_AREA_00	LIN_EXCLUSIVE_AREA_01/ 02	LIN_EXCLUSIVE_AREA_03	LIN_EXCLUSIVE_AREA_04/05	LIN_EXCLUSIVE_AREA_06/07	LIN_EXCLUSIVE_AREA_08	LIN_EXCLUSIVE_AREA_09	LIN_EXCLUSIVE_AREA_10->22	LIN_EXCLUSIVE_AREA_23/24	LIN_EXCLUSIVE_AREA_25
AREA_23/24										
LIN_EXCLUSIVE_AREA_25	x	x	x	x	x	x	x	x	x	

5.2 Peripheral Hardware Requirements

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

5.3 ISR to configure within OS – dependencies

The interrupt service routines are implemented using ISR macro.

The ISR macro implementation depends on the MCAL environment used:

1. Without OS and INTC used in software vector mode:

#define ISR(IsrName) void IsrName(void)

2. Without OS and INTC used in hardware vector mode:

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

3. With Freescale OS:

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

The following ISR's are used by the LIN driver:

Table 5-2. LIN ISR's For LPUART/ FLEXIO channel

ISR Name	Hardware interrupt vector
Lin_LPUART_Isr_LPUART_0	31
Lin_LPUART_Isr_LPUART_1	33 (For S32K118 is 30)
Lin_LPUART_Isr_LPUART_2	35
FlexIO_0_Isr	69 (For S32K118 is 25)

For each interrupts name is added a prefix (ISR) that depends by OS:

- For MCAL without OS, ISR() macro is defined as “OSISR_”.
- For Freescale OS, the ISR() macro is defined as “OS_isr_”.
- For EB OS, the ISR() macro is defined as “OS_ISR_”.

NOTE:

1. The type number and interrupt vectors of hardware channels are specific for each platform. See the documentation for details.
2. In case of AUTOSAR_OS_NOT_USED and the user wants to used the INTC in software mode, the compiler option "-DUSE_SW_VECTOR_MODE" have to be added to the list of compiler options.
3. NO external wake-up is supported by drivers. To wake-up the core from Standby 0 over LIN channels, the ICU drivers has to be used.

5.4 ISR Macro

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

a. OS is not used - AUTOSAR_OS_NOT_USED is defined:

i. If USE_SW_VECTOR_MODE is defined:

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

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

ii. If USE_SW_VECTOR_MODE is not defined:

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

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

Custom OS is used - AUTOSAR_OS_NOT_USED is not defined

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

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

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

5.5 Other AUTOSAR modules - dependencies

- **Base:** The BASE module contains the common files/definitions needed by all MCAL modules.
- **Mcu:** The MCU driver provides services for basic microcontroller initialization, power down functionality, reset and microcontroller specific functions required by other MCAL software modules. The clocks need to be initialized prior to using the 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.
- **EcuC:** The ECUC module is used for ECU configuration. MCAL modules need ECUC to retrieve the variant information.
- **Det:** The DET module is used for enabling Default error detection. The API function used is Det_ReportError(). The activation/deactivation of Default 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_CheckWakeupEvent() 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.

5.6 Data cache restriction

In the DMA transfer mode, DMA transfers may issue cache coherency problems. To avoid possible coherency issues when D-CACHE is enabled, the user shall ensure that the buffers used as TCD source and destination are allocated in the NON-CACHEABLE area (by means of Memmap).

5.7 User Mode Support

Note: Lin module does not include registers protection. So, it is accessible to all registered in any public mode.

Chapter 6
Main API Requirements

6.1 Main functions calls within BSW scheduler

None.

6.2 API Requirements

Not Applicable.

6.3 Calls to Notification Functions, Callbacks, Callouts

Call-back Notifications:

The LIN Driver uses 3 callback functions which have to be provided by the respective module:

EcuM_ValidateWakeupEvent(), EcuM_SetWakeupEvent() and EcuM_CheckWakeup() have to be provided by the EcuM module.

User Notification

None

Chapter 7
Memory Allocation

7.1 Sections to be defined in Lin_MemMap.h

Table 7-1. Memory Allocation

Section name	Type of section	Description
LIN_START_SEC_CONFIG_DATA_UNSPECIFIED	Configuration Data	Start of Memory Section for Config Data
LIN_STOP_SEC_CONFIG_DATA_UNSPECIFIED	Configuration Data	End of Memory Section for Config Data
LIN_START_SEC_CODE	Code	Start of memory Section for Code
LIN_STOP_SEC_CODE	Code	End of memory Section for Code
LIN_START_SEC_VAR_NO_INIT_8	Variables	Used for variables which have to be aligned to 8 bit. For instance used for variables of size 8 bit or used for composite data types: arrays, structs containing elements of maximum 8 bits. These variables are never cleared and never initialized by start-up code.
LIN_STOP_SEC_VAR_NO_INIT_8	Variables	End of above section.
LIN_START_SEC_VAR_NO_INIT_UNSPECIFIED	Variables	Used for variables, structures, arrays when the SIZE (alignment) does not fit the criteria of 8,16 or 32 bit. These variables are never cleared and never initialized by start-up code.
LIN_STOP_SEC_VAR_NO_INIT_UNSPECIFIED	Variables	End of above section.
LIN_START_SEC_VAR_INIT_8	Variables	Used for variables which have to be aligned to 8 bit. For instance used for variables of size 8 bit 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 above section.
LIN_START_SEC_VAR_INIT_UNSPECIFIED	Variables	Used for variables, structures, arrays, when the SIZE (alignment) does not fit the criteria

Table continues on the next page...

Table 7-1. Memory Allocation (continued)

Section name	Type of section	Description
		of 8,16 or 32 bit. These variables are initialized with values after every reset.
LIN_STOP_SEC_VAR_INIT_UNSPECIFIED	Variables	End of above section.
LIN_START_SEC_CONST_32	Constant Data	Used for constants that have to be aligned to 32 bit.
LIN_STOP_SEC_CONST_32	Constant Data	End of above section.
LIN_START_SEC_CONST_UNSPECIFIED	Constant Data	Used for constants when the SIZE (alignment) does not fit the criteria of 8,16 or 32 bit.
LIN_STOP_SEC_CONST_UNSPECIFIED	Constant Data	End of above section.

7.2 Linker command file

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

Chapter 8
Configuration parameters considerations

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

8.1 Configuration Parameters

Specifies whether the configuration parameter shall be of configuration class Post Build

Table 8-1. Configuration Parameters

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
Lin	IMPLEMENTATION_CONFIG_VARIANT	Pre Compile parameter for all Variants of Configuration	Pre compile
NonAutosar			
	LinDisableDemReportErrorStatus	Pre Compile parameter for all Variants of Configuration	Pre compile
	LinFrameTimeoutDisable	Pre Compile parameter for all Variants of Configuration	Pre compile
LinGeneral			
	LinDevErrorDetect	Pre Compile parameter for all Variants of Configuration	Pre compile
	LinIndex	Pre Compile parameter for all Variants of Configuration	Pre compile
	LinTimeoutDuration	Pre Compile parameter for all Variants of Configuration	Pre compile
	LinVersionInfoApi	Pre Compile parameter for all Variants of Configuration	Pre compile
LinChannel			
	LinChannelId	Pre Compile parameter for all Variants of Configuration	Pre compile
	LinHwChannel	Pre Compile parameter for all Variants of Configuration	Pre compile
	LinClockRef	VariantPC or VariantPB	VariantPC or VariantPB
	LinClockRef_Alternate	VariantPC or VariantPB	VariantPC or VariantPB

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
	LinChannelBaudRate	VariantPC or VariantPB	VariantPC or VariantPB
	BreakLength	VariantPC or VariantPB	VariantPC or VariantPB
	LinChannelWakeupSupport	Pre Compile parameter for all Variants of Configuration	Pre compile
	LinChannelEcuMWakeupSource	Pre Compile parameter for all Variants of Configuration	Pre compile
LinDemEventParameterRefs			
	LIN_E_TIMEOUT	Pre Compile parameter for all Variants of Configuration	Pre compile
LinFlexIOModuleConfiguration			
	LinFlexIOEnabledInDebug	Pre Compile parameter for all Variants of Configuration	Pre compile
	LinFlexIOFastAccessMode	Pre Compile parameter for all Variants of Configuration	Pre compile
	LinFlexIOEnabledInDozeMode	Pre Compile parameter for all Variants of Configuration	Pre compile

Chapter 9
Integration Steps

This section gives a brief overview of the steps needed for integrating Local Interconnect Network :

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

Chapter 10
ISR Reference

ISR functions exported by the LIN driver.

10.1 Software specification

The following sections contains driver software specifications.

10.1.1 Define Reference

Constants supported by the driver are as per AUTOSAR LIN Driver software specification Version 4.2 Rev0002 .

10.1.2 Enum Reference

Enumeration of all constants supported by the driver are as per AUTOSAR LIN Driver software specification Version 4.2 Rev0002 .

10.1.3 Function Reference

Functions of all functions supported by the driver are as per AUTOSAR LIN Driver software specification Version 4.2 Rev0002 .

10.1.3.1 Function Lin_LPUART_IsrTxRx_LPUART_0

Interrupt handler on LPUART_0.

Details:

This function implements the ISR occurring on transmission and reception on LPUART_0.

Note

This interrupt handlers is only present if Hardware Channel 0 is used and DMA is not used.

Prototype: `void Lin_LPUART_IsrTxRx_LPUART_0(void);`

10.1.3.2 Function Lin_LPUART_IsrError_LPUART_0

Error interrupt on LPUART_0.

Details:

This function implements the ISR occurring on error on LPUART_0.

Violates: Precautions shall be taken in order to prevent the contents of a header file being included twice.

Note

This interrupt handlers is only present if Hardware Channel 0 is used and DMA is not used.

Prototype: `void Lin_LPUART_IsrError_LPUART_0(void);`

10.1.3.3 Function Lin_LPUART_IsrTxRx_LPUART_1

Interrupt handler on LPUART_1.

Details:

This function implements the ISR occurring on transmission and reception on LPUART_1.

Note

This interrupt handlers is only present if Hardware Channel 1 is used and DMA is not used.

Prototype: `void Lin_LPUART_IsrTxRx_LPUART_1(void);`

10.1.3.4 Function Lin_LPUART_IsrError_LPUART_1

Error interrupt on LPUART_1.

Details:

This function implements the ISR occurring on error on LPUART_1.

Violates: Precautions shall be taken in order to prevent the contents of a header file being included twice.

Note

This interrupt handlers is only present if Hardware Channel 1 is used and DMA is not used.

Prototype: `void Lin_LPUART_IsrError_LPUART_1(void);`

10.1.3.5 Function Lin_LPUART_IsrTxRx_LPUART_2

Interrupt handler on LPUART_2.

Details:

This function implements the ISR occurring on transmission and reception on LPUART_2.

Note

This interrupt handlers is only present if Hardware Channel 2 is used and DMA is not used.

Prototype: `void Lin_LPUART_IsrTxRx_LPUART_2(void);`

10.1.3.6 Function Lin_LPUART_IsrError_LPUART_2

Error interrupt on LPUART_2.

Details:

This function implements the ISR occurring on error on LPUART_2.

Violates: Precautions shall be taken in order to prevent the contents of a header file being included twice.

Note

This interrupt handlers is only present if Hardware Channel 2 is used and DMA is not used.

Prototype: `void Lin_LPUART_IsrError_LPUART_2(void);`

10.1.3.7 Function MCL_FLEXIO_ISR

Interrupt handler on FLEXIO_0.

Details:

This function implements the ISR occurring on transmission and reception on FLEXIO_0.

Note

This interrupt handlers is only present if Hardware Channel 0 is used and DMA is not used.

Prototype: `void MCL_FLEXIO_ISR(void);`

10.1.4 Structs Reference

Data structures supported by the driver are as per AUTOSAR LIN Driver software specification Version 4.2 Rev0002 .

10.1.5 Types Reference

Types supported by the driver are as per AUTOSAR LIN Driver software specification Version 4.2 Rev0002 .

10.1.6 Variables Reference

Variables supported by the driver are as per AUTOSAR LIN Driver software specification Version 4.2 Rev0002 .

Chapter 11
External Assumptions for LIN driver

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

[SMCAL_CPR_EXT51]

<< 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_CheckWakeup()
- Lin_GoToSleep()
- Lin_GoToSleepInternal()
- Lin_Wakeup()
- Lin_GetStatus()
- Lin_SendFrame()

[SMCAL_CPR_EXT52]

<< The application shall call the function Lin_Init only once during runtime, as stated in LIN146. >>

[SMCAL_CPR_EXT53]

<< 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(). >>

[SMCAL_CPR_EXT54]

<< 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.

[SMCAL_CPR_EXT163]

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

[SWS_Lin_00045]

<< One LIN driver provides access to one LIN hardware unit type (simple UART or dedicated LIN hardware) that may consist of several LIN channels. >>

[SWS_Lin_00242]

<< The types Lin_PduType and Lin_StatusType used by LIN driver shall be declared in Lin_GeneralTypes.h . >>

[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 response 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]

<< The upper layer of the LIN Driver has to keep the buffer data consistent until return of function call. >>

[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(). >>

[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. >>

[SWS_Lin_00230]

<< Name: Lin_FrameResponseType

Type: Enumeration

Range: LIN_MASTER_RESPONSE Response is generated from this (master) node

LIN_SLAVE_RESPONSE Response is generated from a remote slave node

LIN_SLAVE_TO_SLAVE Response is generated from one slave to another slave, for the master the response will be anonymous, it does not have to receive the response.

Description: This type is used to specify whether the frame processor is required to transmit the response part of the LIN frame. >>

[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. >>

[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. >>

[SWS_Lin_00233]

<< Name: Lin_StatusType

Type: Enumeration

Range: LIN_NOT_OK LIN frame operation return value.

LIN_TX_BUSY Development or production error occurred

LIN_TX_OK LIN frame operation return value.

LIN_TX_BUSY Successful transmission.

LIN_TX_BUSY LIN frame operation return value.

Ongoing transmission (Header or Response).

LIN_TX_HEADER_ERROR 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_RX_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 LIN 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(). >>

[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. >>

[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_RX_ERROR , LIN_RX_NO_RESPONSE and LIN_OPERATIONAL are sub-states of the channel state LIN_CH_OPERATIONAL. >>

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

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

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

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

Document Number IM2LINASR4.2 Rev0002R1.0.2
Revision 1.0