Integration Manual

for MPC574XG LIN Driver

Document Number: IM35LINASR4.2 Rev0002R1.0.0
Rev. 1.0

Contents

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

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0	17/02/2017	Cuong Le - B53882	Calypso ASR 4.2.2 RTM 1.0.0 release

Chapter 2
Introduction

This integration manual describes the integration requirements for LIN Driver for MPC574XG microcontrollers.

2.1 Supported Derivatives

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

Table 2-1. MPC574XG Derivatives

NXP Semiconductor	MPC5748G_LQFP176, MPC5748G_MAPBGA256, MPC5748G_MAPBGA324, MPC5747G_LQFP176, MPC5747G_MAPBGA256, MPC5747G_MAPBGA324, MPC5746G_LQFP176, MPC5746G_MAPBGA256, MPC5746G_MAPBGA324, MPC5748C_LQFP176, MPC5748C_MAPBGA256, MPC5748C_MAPBGA324, MPC5747C_LQFP176, MPC5747C_MAPBGA256, MPC5747C_MAPBGA324, MPC5746C_LQFP176, MPC5746C_MAPBGA256, MPC5746C_MAPBGA324, MPC5746C_MAPBGA100, MPC5745C_LQFP176, MPC5745C_MAPBGA256, MPC5745C_MAPBGA100, MPC5744C_LQFP176, MPC5744C_MAPBGA256, MPC5744C_MAPBGA100, MPC5746B_LQFP176, MPC5746B_MAPBGA256, MPC5746B_MAPBGA100, MPC5744B_LQFP176, MPC5744B_MAPBGA256,

Table 2-1. MPC574XG Derivatives

	MPC5744B_MAPBGA100, MPC5745B_LQFP176, MPC5745B_MAPBGA256, MPC5745B_MAPBGA100

All of the above microcontroller devices are collectively named as MPC574XG .

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	Development 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
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	AUTOSAR 4.2 Rev0002LIN Driver Software Specification Document.	4.2.2
2	MPC5748G Reference Manual	Rev. 5, 12/2016
3	MPC5746C Reference Manual	Rev. 4, 12/2016
4	MPC5748G_1N81M_Rev.2 (official document) (1N81M)	Jun-16
5	MPC5748G_1N81M_0N78S_Comparison_Summary_v2_0 (internal document) (1N81M, 0N78S)	31.10.2016
6	MPC5746C_1N06M_Rev.4 (official document) (1N06M)	Jul-16
7	MPC5746C_cut1.1_cut2.0_cut2.1_comparison_v0 (internal document) (1N06M, 0N84S, 1N84S)	14-Sep-16
8	C3M_cut2.1_new_errata_20170113 (internal document) (1N84S)	13-Jan-17

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 SemiconductorMPC574XG . It also explains the EB Tresos Studio plugin setup procedure.

3.1 Build Options

The LIN driver files are compiled using

- Windriver DIAB DIAB_5_9_6_2
- Green Hills Multi 7.1.4 / Compiler 2015.1.6

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

Note

The TS_T2D35M10I0R0 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 = 2 identifies PA architecture and Derivative_Id = 35 identifies the MPC574XG)

3.1.1 DIAB Compiler/Linker/Assembler Options

Table 3-1. Compiler Options

Option	Description
-tPPCE200Z4204N3VEN:simple	Sets target processor to PPCE200Z4204N3VEN, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-tPPCE200Z210N3VEN:simple	Sets target processor to PPCE200Z210N3VEN, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-Xdialect-ansi	Follow the ANSI C standard with some additions
-XO	Enables extra optimizations to produce highly optimized code
-g3	Generate symbolic debugger information and do all optimizations.
-Xsize-opt	Optimize for size rather than speed when there is a choice
-Xsmall-data=0	Set Size Limit for 'small data' Variables to zero.
-Xsmall-const=0	Set Size Limit for “small const” Variables to zero.
-Xaddr-sconst=0x11	Specify addressing for constant static and global variables with size less than or equal to -Xsmall-const to far-absolute.
-Xaddr-sdata=0x11	Specify addressing for non-constant static and global variables with size less than or equal to -Xsmall-data in size to far-absolute.
-Xno-common	Disable use of the 'COMMON' feature so that the compiler or assembler will allocate each uninitialized public variable in the .bss section for the module defining it, and the linker will require exactly one definition of each public variable
-Xnested-interrupts	Allow nested interrupts
-Xdebug-dwarf2	Generate symbolic debug information in dwarf2 format
-Xdebug-local-all	Force generation of type information for all local variables
-Xdebug-local-cie	Create common information entry per module
-Xdebug-struct-all	Force generation of type information for all typedefs, struct, union and class types
-Xforce-declarations	Generates warnings if a function is used without a previous declaration
-ee1481	Generate an error when the function was used before it has been declared
-Xmacro-undefined-warn	Generates a warning when an undefined macro name occurs in a #if preprocessor directive
-Xlink-time-lint	Enable the checking of object and function declarations across compilation units, as well as the consistency of compiler options used to compile source files
-W:as:,-l	Pass the option '-l' (lower case letter L) to the assembler to get an assembler listing file
-Wa,-Xisa-vle	Instruct the assembler to expect and assemble VLE (Variable Length Encoding) instructions rather than BookE instructions.
-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
-DUSE_SW_VECTOR_MODE	-D defines a preprocessor symbol and optionally can set it to a value. USE_SW_VECTOR_MODE: By default in the package, drivers are compiled to be used with interrupt controller configured to be in hardware vector mode. In case of AUTOSAR_OS_NOT_USED, the compiler option "-DUSE_SW_VECTOR_MODE" must be added to the list of compiler options to be used with interrupt controller configured to be in software vector mode.

Table continues on the next page...

Table 3-1. Compiler Options (continued)

Option	Description
-DDIAB	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the DIAB preprocessor symbol.
-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.
-c	Stop after assembly, produce object file.

Table 3-2. Assembler Options

Option	Description
-tPPCE200Z4204N3VEN:simple	Sets target processor to PPCE200Z4204N3VEN, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-tPPCE200Z210N3VEN:simple	Sets target processor to PPCE200Z210N3VEN, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-g	Dump the symbols in the global symbol table in each archive file.
-Xisa-vle	Expect and assemble VLE (Variable Length Encoding) instructions rather than Book E instructions. The default code section is named .text_vle instead of .text, and the default code section fill "character" is set to 0x44444444 instead of 0. The .text_vle code section will have ELF section header flags marking it as VLE code, not Book E code.
-Xasm-debug-on	Generate debug line and file information
-Xdebug-dwarf2	Generate symbolic debug information in dwarf2 format
-Xsemi-is-newline	Treat the semicolon (;) as a statement separator instead of a comment character.

Table 3-3. Linker Options

Option	Description
-tPPCE200Z4204N3VEN:simple	Sets target processor to tPPCE200Z4204N3VEN, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-tPPCE200Z210N3VEN:simple	Sets target processor to tPPCE200Z210N3VEN, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-Xelf	Generates ELF object format for output file
-m6	Generates a detailed link map and cross reference table
-Xlink-time-lint	Enable the checking of object and function declarations across compilation units, as well as the consistency of compiler options used to compile source files

3.1.2 GHS Compiler/Linker/Assembler Options

Table 3-4. Compiler Options

Option	Description
-cpu=ppc5748gz4204	Selects target processor: ppc5748gz4204
-cpu=ppc5748gz210	Selects target processor: ppc5748gz210
-ansi	Specifies ANSI C with extensions. This mode extends the ANSI X3.159-1989 standard with certain useful and compatible constructs.
-noSPE	Disables the use of SPE and vector floating point instructions by the compiler.
-Ospace	Optimize for size.
-sda=0	Enables the Small Data Area optimization with a threshold of 0.
-vle	Enables VLE code generation
-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
--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
--diag_error 223	Sets the specified compiler diagnostic messages to the level of error
-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
-DUSE_SW_VECTOR_MODE	-D defines a preprocessor symbol and optionally can set it to a value. USE_SW_VECTOR_MODE: By default in the package, drivers are compiled to be used with interrupt controller configured to be in hardware vector mode. In case of AUTOSAR_OS_NOT_USED, the compiler option "-DUSE_SW_VECTOR_MODE" must be added to the list of compiler options to be used with interrupt controller configured to be in software vector mode.
-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.
-c	Produces an object file (called input-file.o) for each source file.

Table 3-5. Assembler Options

Option	Description
-cpu=ppc5748gz4204	Selects target processor: ppc5748gz4204
-cpu=ppc5748gz210	Selects target processor: ppc5748gz210
-G	Generates source level debugging information and allows procedure call from debugger's command line.
-list	Creates a listing by using the name of the object file with the .lst extension

Table 3-6. Linker Options

Option	Description
-cpu=ppc5748gz4204	Selects target processor: ppc5748gz4204
-cpu=ppc5748gz210	Selects target processor: ppc5748gz210
-nostartfiles	Do not use Start files.
-vle	Enables VLE code generation
--nocpp	Do not Generate Constructors/Destructors
-Mn	sort numerically the MAP file
-delete	The -delete option instructs the linker to remove functions that are not referenced in the final executable.
-ignore_debug_references	Ignores relocations from DWARF debug sections when using -delete. DWARF debug information will contain references to deleted functions that may break some third-party debuggers.
-keepmap	keeps the MAP file in case of link error

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 MPC574XG 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_T2D35M10I0R0\src\Lin.c
- ..\Lin_TS_T2D35M10I0R0\src\Lin_IPW.c
- ..\ Lin_TS_T2D35M10I0R0\src\Lin_LINFlex.c
- ..\ Lin_TS_T2D35M10I0R0\src\Lin_LINFlex_Irq.c
- ..\ Lin_TS_T2D35M10I0R0\src\Lin_NonASR.c

- ..\Lin_TS_T2D35M10I0R0\include\Lin.h
- ..\Lin_TS_T2D35M10I0R0\include\Lin_NonASR.h

- ..\Lin_TS_T2D35M10I0R0\include\Lin_IPW.h
- ..\ Lin_TS_T2D35M10I0R0\include\Lin_LINFlex.h
- ..\ Lin_TS_T2D35M10I0R0\include\Reg_eSys_LINFlex.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 parameters that are not variant aware

Files from Base common folder

- ..\Base_TS_T2D35M10I0R0\generate_PC\include\modules.h
- ..\Base_TS_T2D35M10I0R0\generate_PC\include\Soc_Ips.h
- ..\Base_TS_T2D35M10I0R0\include\Cer.h
- ..\Base_TS_T2D35M10I0R0\include\Compiler.h
- ..\Base_TS_T2D35M10I0R0\include\Compiler_Cfg.h
- ..\Base_TS_T2D35M10I0R0\include\ComStack_Cfg.h
- ..\Base_TS_T2D35M10I0R0\include\ComStack_Types.h
- ..\Base_TS_T2D35M10I0R0\include\Lin_GeneralTypes.h
- ..\Base_TS_T2D35M10I0R0\include\Mcal.h
- ..\Base_TS_T2D35M10I0R0\include\Lin_MemMap.h
- ..\Base_TS_T2D35M10I0R0\include\Platform_Types.h
- ..\Base_TS_T2D35M10I0R0\include\Reg_eSys.h
- ..\Base_TS_T2D35M10I0R0\include\Reg_Macros.h
- ..\Base_TS_T2D35M10I0R0\include\Std_Types.h

Files from Dem folder:

- ..\Dem_TS_T2D35M10I0R0\generate_PC\include\Dem_IntErrId.h
- ..\Dem_TS_T2D35M10I0R0\include\Dem.h
- ..\Dem_TS_T2D35M10I0R0\include\Dem_Types.h

Files from Det folder:

- ..\Det_TS_T2D35M10I0R0\include\Det.h

Files from EcuM folder:

- ..\EcuM_TS_T2D35M10I0R0\generate_PC\include\EcuM_Cfg.h
- ..\EcuM_TS_T2D35M10I0R0\include\EcuM.h
- ..\EcuM_TS_T2D35M10I0R0\include\EcuM_Cbk.h

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 21.0.0 b160607-0933 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_T2D35M10I0R0\config\Base.xdm
  - ..\EcuM_TS_T2D35M10I0R0\config\EcuM.xdm
  - ..\EcuC_TS_T2D35M10I0R0\config\EcuC.xdm
  - ..\Lin_TS_T2D35M10I0R0\config\Lin.xdm
  - ..\Mcu_TS_T2D35M10I0R0\config\Mcu.xdm
  - ..\Resource_TS_T2D35M10I0R0\config\Resource.xdm
- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - ..\Base_TS_T2D35M10I0R0\autosar\Base.epd
  - ..\EcuM_TS_T2D35M10I0R0\autosar\EcuM.epd
  - ..\EcuC_TS_T2D35M10I0R0\autosar\EcuC.epd
  - ..\Lin_TS_T2D35M10I0R0\autosar\Lin.epd
  - ..\Mcu_TS_T2D35M10I0R0\autosar\Mcu.epd
  - ..\Resource_TS_T2D35M10I0R0\autosar\Resource.epd
- Code Generation Templates for parameters without variation points:
  - ..\Lin_TS_T2D35M10I0R0generate_PC\src\Lin_Cfg.c
  - ..\Lin_TS_T2D35M10I0R0generate_PC\include\Lin_Cfg.h
  - ..\Lin_TS_T2D35M10I0R0generate_PC\Lin_VersionCheck_Inc.m
  - ..\Lin_TS_T2D35M10I0R0generate_PC\Lin_VersionCheck_Src.m
  - ..\Lin_TS_T2D35M10I0R0generate_PC\Lin_BaudRate_Comp.m
- Code Generation Templates for variant aware parameters:
  - ..\Lin_TS_T2D35M10I0R0generate_PB\src\Lin_PBcfg.c
  - ..\Lin_TS_T2D35M10I0R0generate_PB\Lin_VersionCheck_Inc.m

- ..\Lin_TS_T2D35M10I0R0generate_PB\Lin_VersionCheck_Src_PB.m
- ..\Lin_TS_T2D35M10I0R0generate_PB\Lin_BaudRate_Comp.m

Steps to generate the configuration:

1. Copy the module folders Lin_TS_T2D35M10I0R0, Base_TS_T2D35M10I0R0,Resource_TS_T2D35M10I0R0, EcuM_TS_T2D35M10I0R0, EcuC_TS_T2D35M10I0R0,Mcu_TS_T2D35M10I0R0 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: MPC5748G_LQFP176, MPC5748G_MAPBGA256, MPC5748G_MAPBGA324, MPC5747G_LQFP176, MPC5747G_MAPBGA256, MPC5747G_MAPBGA324, MPC5746G_LQFP176, MPC5746G_MAPBGA256, MPC5746G_MAPBGA324, MPC5748C_LQFP176, MPC5748C_MAPBGA256, MPC5748C_MAPBGA324, MPC5747C_LQFP176, MPC5747C_MAPBGA256, MPC5747C_MAPBGA324, MPC5746C_LQFP176, MPC5746C_MAPBGA256, MPC5746C_MAPBGA324, MPC5746C_MAPBGA100, MPC5745C_LQFP176, MPC5745C_MAPBGA256, MPC5745C_MAPBGA100, MPC5744C_LQFP176, MPC5744C_MAPBGA256, MPC5744C_MAPBGA100, MPC5746B_LQFP176, MPC5746B_MAPBGA256, MPC5746B_MAPBGA100, MPC5744B_LQFP176, MPC5744B_MAPBGA256, MPC5744B_MAPBGA100, MPC5745B_LQFP176, MPC5745B_MAPBGA256, MPC5745B_MAPBGA100 .
- **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 development 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 LINFlex controllers on chip.
2. **Lin.LinGlobalConfig.LinChannel.LinHwChannel** - list of available LinFlex controllers on chip (LinHWCh_0, LinHWCh_1, etc).

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 LINFlex 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.

Note

After the initialization of the platform, the WUF bit in LINSR register is set. This is the normal behaviour of the chip. The user is responsible to clear this bit, before initializing the LIN driver.

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 LINFlex channel the ISR "LINFlex_Tx_Rx_InterruptHandler(*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 LINFLEX_LINCR1 register during the read/modify/write action. It is used in the function Lin_Linflex_CheckWakeup.

LIN_EXCLUSIVE_AREA_01 To protect the LINFLEX_LINCR1 register during the read/modify/write action. It is used in the function Lin_Linflex_InitChannel.

LIN_EXCLUSIVE_AREA_02 To protect the LINFLEX_LINCR1, LINFLEX_LINTCSR registers during the read/modify/write action. It is used in the function Lin_Linflex_InitChannel.

LIN_EXCLUSIVE_AREA_03 To protect the LINFLEX_LINCR2 register during the read/modify/write action. It is used in the function Lin_Linflex_SendHeader.

LIN_EXCLUSIVE_AREA_04 To protect the LINFLEX_LINTCSR, LINFLEX_LINIER registers during the read/modify/write action. It is used in the function Lin_Linflex_SendHeader.

LIN_EXCLUSIVE_AREA_05 To protect the LINFLEX_LINTCSR, LINFLEX_LINIER, LINFLEX_LINCR2 registers during the read/modify/write action. It is used in the function Lin_Linflex_SendResponse.

LIN_EXCLUSIVE_AREA_06 To protect the LINFLEX_LINTCSR, LINFLEX_LINIER, LINFLEX_LINCR2 registers during the read/modify/write action. It is used in the function Lin_Linflex_GoToSleep.

LIN_EXCLUSIVE_AREA_07 To protect the LINFLEX_LINTCSR, LINFLEX_LINIER, LINFLEX_LINCR2 registers during the read/modify/write action. It is used in the function Lin_Linflex_GoToSleep.

LIN_EXCLUSIVE_AREA_08 To protect the LINFLEX_LINTCSR, LINFLEX_LINIER, LINFLEX_LINCR2, LINFLEX_LINCR1 registers during the read/ modify/write action. It is used in the function Lin_Linflex_GoToSleepInternal.

LIN_EXCLUSIVE_AREA_09 To protect the LINFLEX_LINCR1, LINFLEX_LINCR2 registers during the read/modify/write action. It is used in the function Lin_Linflex_WakeUp.

LIN_EXCLUSIVE_AREA_10 To protect the LINFLEX_LINCR1 register during the read/modify/write action. It is used in the function Lin_Linflex_WakeUpInternal.

LIN_EXCLUSIVE_AREA_11 To protect the LINFLEX_LINTCSR register during the read/modify/write action. It is used in the function Lin_Linflex_TxRxInterruptHandler.

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

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

LIN_EXCLUSIVE_AREA_14 To protect the LINFLEX_LINIER, LINFLEX_LINCR1 registers during the read/modify/write action. It is used in the function Lin_Linflex_TxRxInterruptHandler.

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

LIN_EXCLUSIVE_AREA_16 To protect the LINFLEX_LINCR2 register during the read/modify/write action. It is used in the function Lin_Linflex_ErrorInterruptHandler.

LIN_EXCLUSIVE_AREA_17 To protect the LINFLEX_LINIER, LINFLEX_LINTCSR registers during the read/modify/write action. It is used in the function Lin_Linflex_ErrorInterruptHandler.

LIN_EXCLUSIVE_AREA_18LINFLEX_LINIER To protect the LINFLEX_LINIER register during the read/modify/write action. It is used in the function Lin_Linflex_ErrorInterruptHandler.

LIN_EXCLUSIVE_AREA_19 To protect the LINFLEX_LINIER, LINFLEX_LINCR1 registers during the read/modify/write action. It is used in the function Lin_Linflex_ErrorInterruptHandler.

LIN_EXCLUSIVE_AREA_20 To protect the LINFLEX_LINCR1 registers during the read/modify/write action. It is used in the function Lin_SetClockMode.

LIN_EXCLUSIVE_AREA_21 To protect the LINFLEX_LINCR1 registers during the read/modify/write action. It is used in the function Lin_SetClockMode.

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 5-1. LIN ISR's For STM channel

ISR Name	Hardware interrupt vector
Lin_Linflex_IsrRx_LINFlex_0	376
Lin_Linflex_IsrTx_LINFlex_0	377
Lin_Linflex_IsrError_LINFlex_0	378
Lin_Linflex_IsrRx_LINFlex_1	379
Lin_Linflex_IsrTx_LINFlex_1	380
Lin_Linflex_IsrError_LINFlex_1	381
Lin_Linflex_IsrRx_LINFlex_2	382
Lin_Linflex_IsrTx_LINFlex_2	383
Lin_Linflex_IsrError_LINFlex_2	384
Lin_Linflex_IsrRx_LINFlex_3	385
Lin_Linflex_IsrTx_LINFlex_3	386

Table continues on the next page...

Table 5-1. LIN ISR's For STM channel (continued)

ISR Name	Hardware interrupt vector
Lin_Linflex_IsrError_LINFlex_3	387
Lin_Linflex_IsrRx_LINFlex_4	388
Lin_Linflex_IsrTx_LINFlex_4	389
Lin_Linflex_IsrError_LINFlex_4	390
Lin_Linflex_IsrRx_LINFlex_5	391
Lin_Linflex_IsrTx_LINFlex_5	392
Lin_Linflex_IsrError_LINFlex_5	393
Lin_Linflex_IsrRx_LINFlex_6	394
Lin_Linflex_IsrTx_LINFlex_6	395
Lin_Linflex_IsrError_LINFlex_6	396
Lin_Linflex_IsrRx_LINFlex_7	397
Lin_Linflex_IsrTx_LINFlex_7	398
Lin_Linflex_IsrError_LINFlex_7	399
Lin_Linflex_IsrRx_LINFlex_8	400
Lin_Linflex_IsrTx_LINFlex_8	401
Lin_Linflex_IsrError_LINFlex_8	402
Lin_Linflex_IsrRx_LINFlex_9	403
Lin_Linflex_IsrTx_LINFlex_9	404
Lin_Linflex_IsrError_LINFlex_9	405
Lin_Linflex_IsrRx_LINFlex_10	406
Lin_Linflex_IsrTx_LINFlex_10	407
Lin_Linflex_IsrError_LINFlex_10	408
Lin_Linflex_IsrRx_LINFlex_11	409
Lin_Linflex_IsrTx_LINFlex_11	410
Lin_Linflex_IsrError_LINFlex_11	411
Lin_Linflex_IsrRx_LINFlex_12	412
Lin_Linflex_IsrTx_LINFlex_12	413
Lin_Linflex_IsrError_LINFlex_12	414
Lin_Linflex_IsrRx_LINFlex_13	415
Lin_Linflex_IsrTx_LINFlex_13	416
Lin_Linflex_IsrError_LINFlex_13	417
Lin_Linflex_IsrRx_LINFlex_14	418
Lin_Linflex_IsrTx_LINFlex_14	419
Lin_Linflex_IsrError_LINFlex_14	420
Lin_Linflex_IsrRx_LINFlex_15	421
Lin_Linflex_IsrTx_LINFlex_15	422
Lin_Linflex_IsrError_LINFlex_15	423
Lin_Linflex_IsrRx_LINFlex_16	424
Lin_Linflex_IsrTx_LINFlex_16	425

Table continues on the next page...

Table 5-1. LIN ISR's For STM channel (continued)

ISR Name	Hardware interrupt vector
Lin_Linflex_IsrError_LINFlex_16	426
Lin_Linflex_IsrRx_LINFlex_17	427
Lin_Linflex_IsrTx_LINFlex_17	428
Lin_Linflex_IsrError_LINFlex_17	429

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

- **Det** This module is necessary for enabling Development error detection. The API function used is Det_ReportError(). The activation/deactivation of Development error detection is configurable using 'LinDevErrorDetect' configuration parameter.
- **Dem:** This module is necessary for enabling reporting of production relevant error status. The API function used is Dem_ReportErrorStatus().
- **EcuM:** This module is necessary for a reference to the Wakeup source for this controller as defined in the ECU State Manager.
- **EcuC:** This module is necessary for selecting the postbuild variant criterion.
- **Mcu:** MCU module shall be initialized before using LIN. This module is required for setting the "LIN Clock Reference" value.
- **Port:** For each LINFlex, the TXD and RXD signals need to be configured.
- **Resource:** Sub-Derivative model is selected from Resource configuration.

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 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
	LinDisableFrameTimeout	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

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
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

Document Number IM35LINASR4.2 Rev0002R1.0.0
Revision 1.0