Integration Manual

for MPC5634M LIN Driver

Document Number: IM14LINASR3.0R2.0.0
Rev. 2.0

freescale™

Contents

Section Number **Title** **Page**

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0	07-Feb-2011	Giuseppe Stefano Fazio	Document generation
2.0	14-Dec-2011	Anuj Gupta	Updated for Monaco RTM 2.0.0

Chapter 2 Introduction

This integration manual describes the integration requirements for LIN Driver for MPC5634M microcontrollers.

2.1 Supported Derivatives

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

Table 2-1. MPC5634M Derivatives

Freescale Semiconductor	mpc5634m_bga208, mpc5634m_qfp144, mpc5634m_qfp176

All of the above microcontroller devices are collectively named as MPC5634M .

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

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	AUTOSAR 3.0LIN Driver Software Specification Document.	V2.2.0 R3.0 Rev 0001
2	MPC5634M Reference Manual	Rev. 6, 4 October 2011

Chapter 3
Building the Driver

This section describes the source files and various compilers, linker options used for building the Autosar LIN driver for Freescale SemiconductorMPC5634M . It also explains the EB Tresos Studio plugin setup procedure.

3.1 Build Options

The LIN driver files are compiled using

- GHS 5.2.4
- DIAB 5_8_0_02 wind00198363 20100511 123238
- CW Version 4.3 build 182
- COSMIC Software PPC C Cross Compiler V4.3.4 - 16 Nov 2011 - Win32-F

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

Note

The TS_T2D14M20I0R0 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 PowerPC architecture and Derivative_Id = 14 identifies the MPC5634M)

3.1.1 CW Compiler/Linker/Assembler Options

Table 3-1. Compiler Options

Option	Description
-proc Zen	Generates and links object code for Zen processor. The compiler uses unsigned as the default parameter for the -char switch
-lang c	Expects source code to conform to the language specified by the ISO/IEC 9899-1990 ("C90") standard
-opt all	This option is selected all optimization (the same as -opt speed,level=4,intrinsics,noframe)
-common off	Disables moving uninitialized data into a common section
-sdatathreshold 0	Specifies the threshold size (in bytes) for an item considered by the linker to be small data. (The linker stores small data items in the Small Data address space. The compiler can generate faster code to access this data.)
-sdata2threshold 0	Specifies the threshold size (in bytes) for an item considered by the linker to be small constant data. (The linker stores small constant data items in the Small Constant Data address space.)
-vle	Tells the compiler and linker to generate and lay out Variable Length Encoded (VLE) instructions, available on Zen variants of Power Architecture processors
-use_lmw_stmw on	Enables the use of multiple load and store instructions for function prologues and epilogues
-ir	Include the debug information
-ppc_asm_to_vle	Converts regular Power Architecture assembler mnemonics to equivalent VLE (Variable Length Encoded) assembler mnemonics in the inline assembler
-cpp_exceptions off	When on, generates executable code for C++ exceptions. When off, generates smaller, faster executable code
-func_align 4	Specifies alignment of functions in executable code
-sym dwarf-2,full	Generate DWARF-2-conforming debugging information (Debug With Arbitrary Record Format)
-gdwarf-2	Generate DWARF-2-conforming debugging information (Debug With Arbitrary Record Format). The linker ignores debugging information that is not in the Dwarf 1, Dwarf 2 format
-w on	Turns on most warning messages
-r	Compiler should expect function prototypes
-w undefmacro	Issues warning messages on the use of undefined macros in #if and #elif conditionals
-char unsigned	Controls the default sign of the char data type: char data items are unsigned
-nosyspath	Performs a search of both the user and system paths, treating #include statements of the form #include xyz the same as the form #include "xyz"
-fp none	No floating point code generation
-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
-DEU_DISABLE_ANSILIB_CALLS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the EU_DISABLE_ANSILIB_CALLS preprocessor symbol.
-DMCAL_CER_VALIDATION	-D defines a preprocessor symbol for CER Report

Table continues on the next page...

Table 3-1. Compiler Options (continued)

Option	Description
-DMCAL_VERSION_CHECK	-D defines enable the cross check between the AutoSar component Version Numbers
-DMWERKS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the CWpreprocessor symbol.

Table 3-2. Assembler Options

Option	Description
-proc Zen	Generates and links object code for Zen processor. The compiler uses unsigned as the default parameter for the -char switch
-vle	Tells the compiler and linker to generate and lay out Variable Length Encoded (VLE) instructions, available on Zen variants of Power Architecture processors
-sym dwarf-2,full	Generate DWARF-2-conforming debugging information (Debug With Arbitrary Record Format)
-gdwarf-2	Generate DWARF-2-conforming debugging information (Debug With Arbitrary Record Format). The linker ignores debugging information that is not in the Dwarf 1, Dwarf 2 format.

Table 3-3. Linker Options

Option	Description
-proc Zen	Generates and links object code for Zen processor. The compiler uses unsigned as the default parameter for the -char switch
-code_merging all	Removes duplicated functions to reduce object code size
-far_near_addressing	Simplifies address computations to reduce object code size and improve performance
-vle_enhance_merging	Removes duplicated functions that are called by functions that use VLE instructions to reduce object code size
-listdwarf	DWARF debugging information in the linker’s map file
-sym dwarf-2,full	Generate DWARF-2-conforming debugging information (Debug With Arbitrary Record Format)
-char unsigned	Controls the default sign of the char data type: char data items are unsigned.

3.1.2 DIAB Compiler/Linker/Assembler Options

Table 3-4. Compiler Options

Option	Description
-tPPCE200Z3VEG:simple	Sets target processor to PPCE200Z3, generates ELF using EABI conventions, All Single Hardware Floating Point (Single precision uses hardware, double precision is mapped to single precision), 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
-Xsize-opt	Optimize for size rather than speed when there is a choice

Table continues on the next page...

Table 3-4. Compiler Options (continued)

Option	Description
-Xsmall-data=0	Set Size Limit for “small data” Variables to zero.
-Xsmall-const=0	Set Size Limit for “small const” Variables to zero.
-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
-Xalign-functions=4	Align each function on an address boundary divisible by 4
-g	Generate symbolic debugger information. Do most target-independent optimizations. Also, disable most target-dependent optimizations: option -g2 also disables basic reordering and all peephole optimizations.
-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
-Xforce-prototypes	Generate warnings if a function is used without a previous prototype declaration
-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
-Xlint	Generate warnings when suspicious and non-portable C code is encountered. Enables all warnings
-ei1604	Suppress the warning messages 1604.
-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
-DDIAB	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the DIAB preprocessor symbol.
-DEU_DISABLE_ANSILIB_CALLS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the EU_DISABLE_ANSILIB_CALLS preprocessor symbol.
-DMCAL_CER_VALIDATION	-D defines a preprocessor symbol for CER Report

Table 3-5. Assembler Options

Option	Description
-tPPCE200Z3VEN:simple	Selects target processor: PPCE200Z3, generates ELF using EABI conventions, NO floating point support, 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

Table 3-6. Linker Options

Option	Description
-tPPCE200Z3VEN:simple	Selects target processor: PPCE200Z3, generates ELF using EABI conventions, NO floating point support, 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
-lc	Specifies to linker to search for libc.a
-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.
-Xlibc-old	Enables usage of legacy (pre-release 5.6) libraries

3.1.3 GHS Compiler/Linker/Assembler Options

Table 3-7. Compiler Options

Option	Description
-cpu=ppc563xm	Selects target processor: ppc563xm
-ansi	Enforces strict ANSI mode (C89 standard)
-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.
--no_commons	Allocates uninitialized global variables to a section and initializes them to zero at program startup. This may improve optimizations by giving the compiler optimizer more information about the location of the variable.
-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

Table continues on the next page...

Table 3-7. Compiler Options (continued)

Option	Description
-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
-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
-DGHS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the GHS preprocessor symbol.
-DEU_DISABLE_ANSILIB_CALLS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the EU_DISABLE_ANSILIB_CALLS preprocessor symbol.
-DMCAL_CER_VALIDATION	-D defines a preprocessor symbol for CER Report
-DMCAL_VERSION_CHECK	-D defines enable the cross check between the AutoSar component Version Numbers

Table 3-8. Assembler Options

Option	Description
-cpu=ppc563xm	Selects target processor: ppc563xm

Table 3-9. Linker Options

Option	Description
-cpu=ppc563xm	Selects target processor: ppc563xm
-nostartfiles	Do not use Start files.
-vle	Enables VLE code generation
-linker_warnings	Display linker warnings

3.1.4 CSMC Compiler/Linker/Assembler Options

Table 3-10. Compiler Options

Option	Description
-l	Create listing file; this option directs the compiler to produce an assembly language file with C source line interspersed in it. Please note that the C source lines are commented in the assembly language file: they start with ";".
+modvc	Memory model with "medium size" application, in detail: "data" less than 64kb, "constants" less than 64kb, no code size limit
+rev	Tells the compiler to reverse the order of bits in the bitfields. You need this option in order to use most non-Cosmic header files.
-pc99	authorize the repetition of the const and volatile modifiers in the declaration either directly or indirectly in the typedef.
-pxf	prefix filenames in the debug information with absolute full path name.
+debug	produce debug information to be used by the debug utilities provided with the compiler and by any external debugger.
-DCSMC	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the CSMC preprocessor symbol.
-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
-DEU_DISABLE_ANSILIB_CALLS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the EU_DISABLE_ANSILIB_CALLS preprocessor symbol.
-DMCAL_CER_VALIDATION	-D defines a preprocessor symbol for CER Report
-DMCAL_VERSION_CHECK	-D defines enable the cross check between the AutoSar component Version Numbers

Table 3-11. Assembler Options

Option	Description
-l	create a listing file. The name of the listing file is derived from the input file name by replacing the suffix by the ".ls" extension

Table 3-12. Linker Options

Option	Description
-p	display symbols with physical address instead of logical address in the map file.

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 MPC5634M 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_T2D14M20I0R0\src\Lin.c
- ..\Lin_TS_T2D14M20I0R0\src\Lin_Irq.c
- ..\Lin_TS_T2D14M20I0R0\src\Lin_LLD.c
- ..\Lin_TS_T2D14M20I0R0\src\Lin_NonASR.c
- ..\Lin_TS_T2D14M20I0R0\src\ESCI_LLD.c
- ..\Lin_TS_T2D14M20I0R0\src\Dma_Lin_LLD.c

- ..\Lin_TS_T2D14M20I0R0\include\Lin.h
- ..\Lin_TS_T2D14M20I0R0\include\Lin_NonASR.h
- ..\Lin_TS_T2D14M20I0R0\include\Lin_LLD.h
- ..\Lin_TS_T2D14M20I0R0\include\ESCI_LLD.h
- ..\Lin_TS_T2D14M20I0R0\include\Reg_eSys_ESCI.h
- ..\Lin_TS_T2D14M20I0R0\include\Reg_eSys_DMA.h
- ..\Lin_TS_T2D14M20I0R0\include\Dma_LLD.h

LIN Generated Files

- Lin_PBcfg.c (For PB Variant) - This file should be generated by the user using a configuration tool for compilation (it contains initialization values for Post Build variant).
- Lin_Cfg.c (For PC Variant) - This file should be generated by the user using a configuration tool for compilation (it contains initialization values for Pre Compile variant).
- Lin_Cfg.h (For PC and PB Variants) - This file should be generated by the user using a configuration tool for compilation (common file containing data and structure definition).

Files from Base common folder

- ..\Base_TS_T2D14M20I0R0\include\Cer.h
- ..\Base_TS_T2D14M20I0R0\include\Compiler.h
- ..\Base_TS_T2D14M20I0R0\include\Compiler_Cfg.h
- ..\Base_TS_T2D14M20I0R0\include\ComStack_Types.h
- ..\Base_TS_T2D14M20I0R0\include\MemMap.h
- ..\Base_TS_T2D14M20I0R0\include\Mcal.h
- ..\Base_TS_T2D14M20I0R0\include\Platform_Types.h
- ..\Base_TS_T2D14M20I0R0\include\Std_Types.h
- ..\Base_TS_T2D14M20I0R0\include\Reg_eSys.h
- ..\Base_TS_T2D14M20I0R0\include\Soc_Ips.h
- ..\Base_TS_T2D14M20I0R0\include\Reg_Macros.h

Files from Dem folder:

- ..\Dem_TS_T2D14M20I0R0\include\Dem.h
- ..\Dem_TS_T2D14M20I0R0\include\Dem_IntErrId.h
- ..\Dem_TS_T2D14M20I0R0\include\Dem_Types.h

Files from Det folder:

- ..\Det_TS_T2D14M20I0R0\include\Det.h

Files from EcuM folder:

- ..\EcuM_TS_T2D14M20I0R0\common\include\EcuM.h

Files from LinIf folder:

- ..\LinIf_TS_T2D14M20I0R0\include\LinIf_Cbk.h

3.3 Setting up the Plug-ins

The LIN driver was designed to be configured by using the EB Tresos Studio (version Tresos 2010a.sr4 20100415-release2010a-sr4 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_T2D14M20I0R0\config\Base.xdm
  - ..\EcuM_TS_T2D14M20I0R0\config\EcuM.xdm
  - ..\Lin_TS_T2D14M20I0R0\config\Lin.xdm
  - ..\LinIf_TS_T2D14M20I0R0\config\LinIf.xdm
  - ..\Mcu_TS_T2D14M20I0R0\config\Mcu.xdm
  - ..\Resource_TS_T2D14M20I0R0\config\Resource.xdm

- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - ..\Base_TS_T2D14M20I0R0\autosar\Base.epd
  - ..\EcuM_TS_T2D14M20I0R0\autosar\EcuM.epd
  - ..\Lin_TS_T2D14M20I0R0\autosar\Lin.epd
  - ..\LinIf_TS_T2D14M20I0R0\autosar\LinIf.epd
  - ..\Mcu_TS_T2D14M20I0R0\autosar\Mcu.epd
  - ..\Resource_TS_T2D14M20I0R0\autosar\Resource.epd

- Code Generation Templates for Pre-Compile time configuration parameters:
  - ..\Lin_TS_T2D14M20I0R0generate_PC\src\Lin_Cfg.c
  - ..\Lin_TS_T2D14M20I0R0generate_PC\include\Lin_Cfg.h
  - ..\Lin_TS_T2D14M20I0R0generate_PC\Lin_VersionCheck_Inc.m

  - ..\Lin_TS_T2D14M20I0R0generate_PC\Lin_VersionCheck_Src.m
  - ..\Lin_TS_T2D14M20I0R0generate_PC\Lin_BaudRate_Comp.m

- Code Generation Templates for Post-Build time configuration parameters:
  - ..\Lin_TS_T2D14M20I0R0generate_PB\src\Lin_PBcfg.c
  - ..\Lin_TS_T2D14M20I0R0generate_PB\Lin_VersionCheck_Src_PB.m
  - ..\Lin_TS_T2D14M20I0R0generate_PB\Lin_BaudRate_Comp.m

Steps to generate the configuration:

1. Copy the module folders Lin_TS_T2D14M20I0R0,LinIf_TS_T2D14M20I0R0, Base_TS_T2D14M20I0R0,Resource_TS_T2D14M20I0R0, EcuM_TS_T2D14M20I0R0,Mcu_TS_T2D14M20I0R0 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: mpc5634m_bga208, mpc5634m_qfp144, mpc5634m_qfp176 .
- **LINIF** is required for reporting status of some events.
- **ECUM** is required for selecting the reference to the wakeup source for every Lin controller.
- **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 eSCI controllers on chip.
2. **Lin.LinGlobalConfig.LinChannel.LinHwChannel** - list of available eSCI 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 eSCI 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. For this purposes the Lin_InitChannel(LIN_CHANNEL, <&Lin_ChannelConfiguration>) API service shall be used.

4.2 Function Calls during Shutdown

There is no shutdown specific procedure for Lin driver, however Lin_DeInitChannel(LIN_CHANNEL) API service may be called to reset the channel to a known uninitialized state. 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 eSCI channel the ISR "ESCI_LLD_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_WakeUpValidation() service API in order to identify if a salve 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

None.

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 eSCI channel

ISR Name	Hardware interrupt vector
Lin_LLD_InterruptHandler_ESCI_0	146
Lin_LLD_InterruptHandler_ESCI_1	149
Lin_InterruptFrame_eSci_A_Tx	29
Lin_InterruptFrame_eSci_A_Rx	30

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.
- **LinIf:** This module is required because it contains the definition of the NetworkHandleType type and the implementation of the LinIf_Cbk_CheckWakeup callback which is called by the LIN driver.
- **Mcu:** MCU module shall be initialized before using LIN. This module is required for setting the "LIN Clock Reference" value.
- **Port:** For each eSCI, the TXD and RXD signals need to be configured.
- **Resource:** Sub-Derivative model is selected from Resource configuration.

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 1 call back function which have to be provided by the respective module:

EcuM_SetWakeupEvent() has to be provided by the EcuM module.

User Notification

None

Chapter 7
Memory Allocation

7.1 Sections to be defined in MemMap.h

For Post Build data:

```
         #ifdef LIN_START_CONFIG_DATA_UNSPECIFIED
#undef LIN_START_CONFIG_DATA_UNSPECIFIED
#undef MEMMAP_ERROR
/*Memory Section for Post Build Data to be defined here.
 Example given in the next line*/
 #pragma ghs section const=".pblin_cfg"
#endif
#ifdef LIN_STOP_CONFIG_DATA_UNSPECIFIED
#undef LIN_STOP_CONFIG_DATA_UNSPECIFIED
#undef MEMMAP_ERROR
/*End of section to be mentioned here. Example given in the
next line.*/
#pragma ghs section
#endif
```

For Code:

```
         #ifdef LIN_START_SEC_CODE
#undef LIN_START_SEC_CODE
#undef MEMMAP_ERROR
/*Memory Section for Code to be defined here.*/
#endif
#ifdef LIN_STOP_SEC_CODE
#undef LIN_STOP_SEC_CODE
#undef MEMMAP_ERROR
/*End of section to be mentioned here*/
#endif
```

For Variables:

```
         #ifdef LIN_START_SEC_VAR_UNSPECIFIED
#undef LIN_START_SEC_VAR_UNSPECIFIED
#undef MEMMAP_ERROR
/*Memory Section for Variables to be defined here.*/
#endif
```

```
#ifdef LIN_STOP_SEC_VAR_UNSPECIFIED
#undef LIN_STOP_SEC_VAR_UNSPECIFIED
#undef MEMMAP_ERROR
/*End of section to be mentioned here*/
#endif
```

For Constant data:

```
         #ifdef LIN_START_SEC_CONST_UNSPECIFIED
#undef LIN_START_SEC_CONST_UNSPECIFIED
#undef MEMMAP_ERROR
/*Memory Section for Constants to be defined here.*/
#endif
#ifdef LIN_STOP_SEC_CONST_UNSPECIFIED
#undef LIN_STOP_SEC_CONST_UNSPECIFIED
#undef MEMMAP_ERROR
/*End of section to be mentioned here*/
#endif
```

7.2 Linker command file

Memory shall be allocated for every section defined in 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
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

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
LinChannel	LinChannelId	VariantPC or VariantPB	Compile Time/Post Build
	LinHwChannel	VariantPC or VariantPB	Compile Time/Post Build
	LinClockRef	VariantPC or VariantPB	Compile Time/Post Build
	LinClockRef_Alternate	VariantPC or VariantPB	Compile Time/Post Build
	LinChannelBaudRate	VariantPC or VariantPB	Compile Time/Post Build
	LinChannelWakeUpSupport	VariantPC or VariantPB	Compile Time/Post Build
	LinChannelEcuMWakeUpSou rce	VariantPC or VariantPB	Compile Time/Post Build
	LinChannelDmaActivation	VariantPC or VariantPB	Compile Time/Post Build
	LinChannelMscSupport	VariantPC or VariantPB	Compile Time/Post Build
	ReceiverFullInterrupt Enable	VariantPC or VariantPB	Compile Time/Post Build
	MscParity	VariantPC or VariantPB	Compile Time/Post Build
	MscFramePolarity	VariantPC or VariantPB	Compile Time/Post Build
	MscClockFrequency	VariantPC or VariantPB	Compile Time/Post Build
	MscPollingMode	VariantPC or VariantPB	Compile Time/Post Build
	MscFrameFormat	VariantPC or VariantPB	Compile Time/Post Build
	MscClockDivisor	VariantPC or VariantPB	Compile Time/Post Build
	DMADataSize	VariantPC or VariantPB	Compile Time/Post Build
	nBytesToTransfer	VariantPC or VariantPB	Compile Time/Post Build
	iterationCount	VariantPC or VariantPB	Compile Time/Post Build
	destinationAddressOffset	VariantPC or VariantPB	Compile Time/Post Build
	lastDestinationAddress	VariantPC or VariantPB	Compile Time/Post Build
	TheHalfPointInterrupt	VariantPC or VariantPB	Compile Time/Post Build
	TheEndOfMajorLoopInterrupt	VariantPC or VariantPB	Compile Time/Post Build
LinNonAutosar	LinEnableDualClockMode	VariantPC or VariantPB	Pre Compile

Chapter 9
Integration Steps

This section gives a brief overview of the steps needed for integrating LIN Driver :

- Generate the required LIN configurations. For more details refer to section Files required for Compilation
- Allocate proper memory sections in MemMap.h and linker command file. For more details refer to section Sections to be defined in MemMap.h
- 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 3.0 .

10.1.2 Enum Reference

Enumeration of all constants supported by the driver are as per AUTOSAR LIN Driver software specification Version 3.0 .

10.1.3 Function Reference

Functions of all functions supported by the driver are as per AUTOSAR LIN Driver software specification Version 3.0 .

10.1.3.1 Function Lin_InterruptFrame_eSci_A_Rx

DMA Rx interrupt handler on ESCI_0.

Details:

This function implements the ISR occurring on DMA Rx on ESCI_0

Note

This interrupt handler is only present if Hardware Channel 0 is used with DMA.

Prototype: `void Lin_InterruptFrame_eSci_A_Rx(void);`

10.1.3.2 Function Lin_InterruptFrame_eSci_A_Tx

DMA Tx interrupt handler on ESCI_0.

Details:

This function implements the ISR occurring on DMA Tx on ESCI_0

Note

This interrupt handler is only present if Hardware Channel 0 is used with DMA.

Prototype: `void Lin_InterruptFrame_eSci_A_Tx(void);`

10.1.3.3 Function Lin_LLD_InterruptHandler_ESCI_0

Interrupt handler on ESCI_0.

Details:

This function implements the ISR occurring on ESCI_0

Note

This interrupt handler is only present if Hardware Channel 0 is used (with or without DMA).

Prototype: `void Lin_LLD_InterruptHandler_ESCI_0(void);`

10.1.3.4 Function Lin_LLD_InterruptHandler_ESCI_1

Interrupt handler on ESCI_1.

Details:

This function implements the ISR occurring on ESCI_1

Note

This interrupt handler is only present if Hardware Channel 1 is used (with or without DMA).

Prototype: `void Lin_LLD_InterruptHandler_ESCI_1(void);`

10.1.4 Structs Reference

Data structures supported by the driver are as per AUTOSAR LIN Driver software specification Version 3.0 .

10.1.5 Types Reference

Types supported by the driver are as per AUTOSAR LIN Driver software specification Version 3.0 .

10.1.6 Variables Reference

Variables supported by the driver are as per AUTOSAR LIN Driver software specification Version 3.0 .

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: IM14LINASR3.0R2.0.0
Rev. 2.0