Integration Manual

for S32K1XX BASE Driver

Rev. 1.0 — 11 January 2022
IM2BASEASR4.2 Rev0002R1.0.5

Revision History

Revision	Date	Author	Description
1.0	11/01/2022	NXP MCAL Team	Updated version for ASR 4.2.2 S32K1XX R1.0.5

1 Introduction

This integration manual describes the integration requirements for BASE Driver for S32K1XX microcontrollers.

1.1 Supported Derivatives

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

Table 1. S32K1XX Derivatives

NXP Semiconductors	s32k142_lqfp100, s32k142_lqfp64, s32k142_lqfp48, s32k144_lqfp100, s32k144_mapbga100, s32k144_lqfp64, s32k144_lqfp48, s32k146_lqfp144, s32k146_lqfp100, s32k146_mapbga100, s32k146_lqfp64, s32k148_lqfp176, s32k148_lqfp144, s32k148_lqfp100, s32k148_mapbga100, s32k118_lqfp48, s32k118_lqfp64, s32k116_lqfp48, s32k116_qfn32, s32k144w_lqfp48, s32k144w_lqfp64, s32k142w_lqfp64, s32k142w_lqfp48

All of the above microcontroller devices are collectively named as S32K1XX .

1.2 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.

1.3 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.*

1.4 Acronyms and Definitions

Table 2. Acronyms and Definitions

Term	Definition
API	Application Programming Interface
ASM	Assembler Language
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
N/A	Not Applicable
MCU	Micro Controller Unit
VLE	Variable Length Encoding

1.5 Reference List

Table 3. Reference List

#	Title	Version
1	General Specification of Basic Software Modules	AUTOSAR Release 4.2.2
2	Specification of Communication Stack Types	AUTOSAR Release 4.2.2
3	Specification of Compiler Abstraction	AUTOSAR Release 4.2.2
4	Specification of Platform Types	AUTOSAR Release 4.2.2
5	Specification of Standard Types	AUTOSAR Release 4.2.2
6	S32K1xx Series Reference Manual	Rev. 13, 04/2020
7	S32K1xx Data Sheet	Rev. 13, 04/2020
8	S32K142 Mask Set Errata for Mask 0N33V (0N33V)	Rev. 20/APR/2020
9	S32K144 Mask Set Errata for Mask 0N57U (0N57U)	Rev. 20/APR/2020
10	S32K146 Mask Set Errata for Mask 0N73V (0N73V)	Rev. 20/APR/2020
11	S32K148 Mask Set Errata for Mask 0N20V (0N20V)	Rev. 20/APR/2020
12	S32K118 Mask Set Errata for Mask 0N97V (0N97V)	Rev. 20/APR/2020
13	S32K116 Mask Set Errata for Mask 0N96V (0N96V)	Rev. 20/APR/2020
14	S32K144W Mask Set Errata for Mask 0P64A (0P64A)	Rev. 14 FEB 2020

2 Building the Driver

This section describes the source files and various compilers, linker options used for building the Autosar BASE driver for NXP Semiconductors S32K1XX . It also explains the EB Tresos Studio plugin setup procedure.

2.1 Build Options

The BASE 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 (Thu Dec 7 13:28:42 CST 2017 build.sh rev=g7fea41d s=L631 Earmv7 -V release_g7fea41d_build_Fed_Earmv7
- IAR: V8.11.2

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

Note: *The TS_T40D2M10I5R0 plugin name is composed as follow:*

TS_T = Target_Id

D = Derivative_Id

M = SW_Version_Major and SW_Version_Minor

I = SW_Version_Patch

R = Reserved

(i.e. Target_Id = 40 identifies CORTEXM architecture and Derivative_Id = 2 identifies the S32K1XX)

2.1.1 GHS Compiler/Linker/Assembler Options

Table 4. 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.

Table 4. Compiler Options*...continued*

Option	Description
--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 5. 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 6. 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

Table 6. Linker Options*...continued*

Option	Description
-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

2.1.2 IAR Compiler/Linker/Assembler Options

Table 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=Pa 050	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 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.

Table 8. Assembler Options*...continued*

Option	Description
-g	Use this option to disable the automatic search for system include files.

Table 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.

2.1.3 GCC Compiler/Linker/Assembler Options

Table 10. 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.
-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.

Table 10. Compiler Options*...continued*

Option	Description
-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 11. 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 12. 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.
-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+

Table 12. Linker Options*...continued*

Option	Description
$(TOOLCHAIN_DIR)/arm-none-eabi/newlib/lib/thumb $(TOOLCHAIN_DIR)/arm-none-eabi/newlib/lib)	Library for core M4

2.2 Files required for Compilation

This section describes the include files required to compile, assemble (if assembler code) and link the BASE driver for S32K1XX 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.

BASE Files

- ..\Base_TS_T40D2M10I5R0\include\Can_GeneralTypes.h
- ..\Base_TS_T40D2M10I5R0\include\Compiler.h
- ..\Base_TS_T40D2M10I5R0\include\Compiler_Cfg.h
- ..\Base_TS_T40D2M10I5R0\include\ComStack_Cfg.h
- ..\Base_TS_T40D2M10I5R0\include\ComStack_Types.h
- ..\Base_TS_T40D2M10I5R0\include\Eth_GeneralTypes.h
- ..\Base_TS_T40D2M10I5R0\include\Fr_GeneralTypes.h
- ..\Base_TS_T40D2M10I5R0\include\Lin_GeneralTypes.h
- ..\Base_TS_T40D2M10I5R0\include\Mcal.h
- ..\Base_TS_T40D2M10I5R0\include\MemMap.h
- ..\Base_TS_T40D2M10I5R0\include\Platform_Types.h
- ..\Base_TS_T40D2M10I5R0\include\Reg_eSys.h
- ..\Base_TS_T40D2M10I5R0\include\RegLockMacros.h
- ..\Base_TS_T40D2M10I5R0\include\SilRegMacros.h
- ..\Base_TS_T40D2M10I5R0\include\Soc_Ips.h
- ..\Base_TS_T40D2M10I5R0\include\Std_Types.h
- ..\Base_TS_T40D2M10I5R0\include\StdRegMacros.h

BASE Generated Files

- modules.h - contains a list of defines associated with each MCAL module present in the project. This file is used internally be MCAL and should be final. (every time you add or remove a MCAL module to or from the project, regenerate this file and use the updated version for re-compiling **ALL** MCAL modules).
- Reg_eSys.h - contains a list of defines base address for peripheral. This file is used internally be MCAL and should be final. (every time you add or remove a MCAL module to or from the project, regenerate this file and use the updated version for re-compiling **ALL** MCAL modules).

Given the fact that some of the files from the BASE module are stubs and must be re-written by the integrator, please read the detailed description of each file from the BASE User Manual.

2.3 Setting up the Plug-ins

The BASE 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 BASE module folder:

- VSMD (Vendor Specific Module Definition) file in EB tresos Studio XDM format:
  - ..\Base_TS_T40D2M10I5R0\config\Base.xdm
- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - ..\Base_TS_T40D2M10I5R0\config\Base.epd
- Code Generation Templates for Pre-Compile time configuration parameters:
  - ..\Base_TS_T40D2M10I5R0\generate_PC\include\modules.h

Steps to generate the configuration:

1. Copy the module folders Base_TS_T40D2M10I5R0 , Resource_TS_T40D2M10I5R0 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

- **RESOURCE** is required to select processor derivative.
- **All other MCAL modules needed for the project.** Base generates a header file with defines associated with each MCAL module present in the project. This file is used internally be MCAL and should be final.

3 Function calls to module

3.1 Function Calls during Start-up

BASE has no function that shall be called during STARTUP phase of EcuM initialization.

3.2 Function Calls during Shutdown

BASE has no function that shall be called during Shutdown phase.

3.3 Function Calls during Wake-up

BASE has no function that shall be called during Wake-up phase.

4 Module requirements

4.1 Exclusive areas to be defined in BSW scheduler

None.

4.2 Peripheral Hardware Requirements

None.

4.3 ISR to configure within OS – dependencies

None.

4.4 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.

b. 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.

4.5 Other AUTOSAR modules - dependencies

- **Resource:** Sub-Derivative model is selected from Resource configuration.
- **All other MCAL modules needed for the project.** Base generates a header files with defines associated with each MCAL module present in the project. This file is used internally be MCAL and should be final.

4.6 Data cache restriction

None

4.7 User Mode support

The **Base** module contains the enablement for running MCAL in **USER_MODE** This is done by the define **MCAL_ENABLE_USER_MODE_SUPPORT**.

By default, the MCAL is running in **SUPERVISOR_MODE**.

In order to run the MCAL in **USER_MODE**, the following compiler option must be added: **MCAL_ENABLE_USER_MODE_SUPPORT**

If the **USER_MODE** is active, and no OS is used, the following functions must be defined:

- **uint8 Sys_GoToSupervisor(void)** to switch the chip to SUPERVISOR mode.
- **uint32 Sys_GoToUser_Return(uint32 u32returnValue)** to switch the chip to USER mode and return the value from the user's function.
- **void Sys_GoToUser(void)** to switch the chip to USER mode.

5 Main API Requirements

5.1 Main functions calls within BSW scheduler

None.

5.2 API Requirements

None.

5.3 Calls to Notification Functions, Callbacks, Callouts

None.

6 Memory Allocation

6.1 Sections to be defined in MemMap.h

BASE module contains the definitions of all memory sections (inside the MemMap.h stub file) but BASE does not use any of these memory sections. It only provides them for the other MCAL modules.

6.2 Linker command file

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

7 Integration Steps

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

- Generate the required BASE configurations. For more details refer to section Section 2.2
- Allocate proper memory sections in BASE_MemMap.h and linker command file. For more details refer to section Section 6.1
- Compile & build the BASE with all the dependent modules. For more details refer to section Section 2

8 External Assumptions for BASE driver

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

N/A

9 Legal information

9.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

9.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

9.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 11 January 2022
Document identifier: AUTOSAR_MCAL_BASE_IM
Document number: IM2BASEASR4.2 Rev0002R1.0.5