
Integration Manual

for S32K14X CAN Driver

Document Number: IM2CANASR4.3 Rev0001R1.0.1
Rev. 1.0





Contents

Section number	Title	Page
Chapter 1		
Revision History		
Chapter 2		
Introduction		
2.1	Supported Derivatives.....	7
2.2	Overview.....	7
2.3	About this Manual.....	8
2.4	Acronyms and Definitions.....	8
2.5	Reference List.....	9
Chapter 3		
Building the Driver		
3.1	Build Options.....	11
3.1.1	GHS Compiler/Linker/Assembler Options.....	11
3.1.2	IAR Compiler/Linker/Assembler Options.....	13
3.1.3	GCC Compiler/Linker/Assembler Options.....	14
3.2	Files Required for the Compilation.....	16
3.3	Setting up the Plugins.....	18
Chapter 4		
Function calls to module		
4.1	Function Calls during Start-up.....	21
4.2	Function Calls during Shutdown.....	21
4.3	Function Calls during Wake-up.....	22
Chapter 5		
Module requirements		
5.1	Exclusive areas to be defined in BSW scheduler.....	25
5.2	Peripheral Hardware Requirements.....	28
5.3	ISR to configure within OS – dependencies.....	28
5.4	ISR Macro.....	29

Section number	Title	Page
5.4.1	The following ISR's are used by the CAN driver.....	30
5.4.2	Macros for Interrupts.....	30
5.5	Other AUTOSAR Modules - Dependencies.....	32
5.6	Data Cache Restriction	33
5.7	User Mode Support.....	33

Chapter 6 Main API Requirements

6.1	Main Functions Calls within BSW Scheduler.....	35
6.2	API Requirements.....	36
6.3	Calls to Notification Functions, Callbacks, Callouts.....	36

Chapter 7 Memory Allocation

7.1	Sections to be defined in MemMap.h.....	39
7.2	Linker command file.....	39

Chapter 8 Configuration parameters considerations

8.1	Configuration Parameters.....	41
-----	-------------------------------	----

Chapter 9 Integration Steps

Chapter 10 ISR Reference

10.1	Function Can_IsrFCx_BO.....	49
10.2	Function Can_IsrFCx_ERR.....	49
10.3	Function Can_IsrFCx_MB_00_15.....	49
10.4	Function Can_IsrFCx_MB_16_31.....	50
10.5	Function Can_IsrFCx_RxFifoEventsMbMix.....	50
10.6	Function Can_IsrFCx_UNI.....	50

Chapter 11 External Assumptions for CAN driver

Chapter 1

Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0	21/06/2019	NXP MCAL Team	Updated version for ASR 4.3.1S32K14XR1.0.1



Chapter 2

Introduction

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

2.1 Supported Derivatives

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

Table 2-1. S32K14X Derivatives

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

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

2.2 Overview

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

AUTOSAR

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

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

2.3 About this Manual

This Technical Reference employs the following typographical conventions:

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

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

Notes and warnings are shown as below:

Note

This is a note.

2.4 Acronyms and Definitions

Table 2-2. Acronyms and Definitions

Term	Definition
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
ASM	Assembler
BSMI	Basic Software Make file Interface
CAN	Controller Area Network
DEM	Diagnostic Event Manager
DET	Development Error Tracer
C/CPP	C and C++ Source Code
VLE	Variable Length Encoding
N/A	Not Applicable
MCU	Micro Controller Unit

2.5 Reference List

Table 2-3. Reference List

#	Title	Version
1	Specification of CAN Driver	AUTOSAR Release 4.3.1
2	S32K14X Reference Manual	Reference Manual, Rev. 9, 9/2018
3	S32K142 Mask Set Errata for Mask 0N33V (0N33V)	30/11/2017
4	S32K144 Mask Set Errata for Mask 0N57U (0N57U)	30/11/2017
5	S32K146 Mask Set Errata for Mask 0N73V (0N73V)	30/11/2017
6	S32K148 Mask Set Errata for Mask 0N20V (0N20V)	25/10/2018
7	S32K118 Mask Set Errata for Mask 0N97V (0N97V)	07/01/2019

Chapter 3

Building the Driver

This section describes the source files and various compilers, linker options used for building the Autosar CAN driver for NXP Semiconductors S32K14X. It also explains the EB Tresos Studio plugin setup procedure.

3.1 Build Options

The CAN driver files are compiled using

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

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

Note

The TS_T40D2M10I1R0 plugin name is composed as follow:

TS_T = Target_Id

D = Derivative_Id

M = SW_Version_Major

I = SW_Version_Minor

R = Revision

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

3.1.1 GHS Compiler/Linker/Assembler Options

Table 3-1. Compiler Options

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

Table 3-2. Assembler Options

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

Table 3-3. Linker Options

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

3.1.2 IAR Compiler/Linker/Assembler Options

Table 3-4. 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 endianness 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.

Table continues on the next page...

Table 3-4. Compiler Options (continued)

Option	Description
--diag_suppress=Pa050	Suppresses diagnostic messages (warnings) about non-standard line endings.
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DIAR	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the IAR preprocessor symbol.
--require_prototypes	Forces the compiler to verify that all functions have proper prototypes.
--no_wrap_diagnostics	Disables line wrapping of diagnostic messages issued by compiler.
--no_system_include	Disables the automatic search for system include files.
-e	Enables language extensions. This option is needed by FLS driver which uses _packed structures.

Table 3-5. Assembler Options

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

Table 3-6. Linker Options

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

3.1.3 GCC Compiler/Linker/Assembler Options

Table 3-7. 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	Generates an extra file that specifies the maximum amount of stack used, on a per-function basis.
-fdump-ipa-all	Enables all inter-procedural analysis dumps.
-Werror=implicit-function-declaration	Generates an error when the prototype of the function is not defined..
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DGCC	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the GCC preprocessor symbol.
-std=c99	C programming language standard version c99

Table 3-8. Assembler Options

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

Table 3-9. 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+, added with -L and -B option
\$(TOOLCHAIN_DIR)/arm-none-eabi/newlib/lib/thumb \$(TOOLCHAIN_DIR)/arm-none-eabi/newlib/lib)	Library for core M4, added with -L and -B option

3.2 Files Required for the Compilation

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

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

CAN Files

- ..\Can_TS_T40D2M10I1R0\include\Can.h
- ..\Can_TS_T40D2M10I1R0\include\Can_IPW.h
- ..\Can_TS_T40D2M10I1R0\include\Can_FlexCan.h
- ..\Can_TS_T40D2M10I1R0\include\Reg_eSys_FlexCan.h
- ..\Can_TS_T40D2M10I1R0\src\Can.c
- ..\Can_TS_T40D2M10I1R0\src\Can_Irq.c
- ..\Can_TS_T40D2M10I1R0\src\Can_FlexCan.c

CAN Generated Files

For driver compilation, Can_[VariantName]_PBcfg.c should be generated by the user using a configuration tool. The file contains the definition of the init pointer for the respective variant.

As a deviation from standard:

- Can_[VariantName]_PBcfg.c (For PB Variant) - files will contain the definition for all parameters that are variant aware, independent of the configuration class that will be selected (PC, LT,PB).
- Can_Cfg.c (For PC Variant) - file will contain the definition for all configuration structures containing only variables that are not variant aware, configured and generated only once. This file alone does not contain the whole structure needed by Can_Init function to configure the driver. Based on the number of variants configured in the EcuC, there can be more than one configuration structure for one module even for PreCompile variant.

Files from Base common folder

- ..\Base_TS_T40D2M10I1R0\include\Compiler.h
- ..\Base_TS_T40D2M10I1R0\include\Compiler_Cfg.h
- ..\Base_TS_T40D2M10I1R0\include\ComStack_Types.h
- ..\Base_TS_T40D2M10I1R0\include\MemMap.h
- ..\Base_TS_T40D2M10I1R0\include\Mcal.h
- ..\Base_TS_T40D2M10I1R0\include\Platform_Types.h
- ..\Base_TS_T40D2M10I1R0\include\Std_Types.h
- ..\Base_TS_T40D2M10I1R0\include\Reg_eSys.h
- ..\Base_TS_T40D2M10I1R0\include\Soc_Ips.h
- ..\Base_TS_T40D2M10I1R0\include\Reg_Macros.h
- ..\Base_TS_T40D2M10I1R0\include\Can_GeneralTypes.h

Files from CanIf folder:

- ..\CanIf_TS_T40D2M10I1R0\include\CanIf_Cbk.h
- ..\CanIf_TS_T40D2M10I1R0\include\CanIf_Types.h.h
- ..\CanIf_TS_T40D2M10I1R0\include\CanIf.h

Files from Det folder:

- ..\Det_TS_T40D2M10I1R0\src\Det.c
- ..\Det_TS_T40D2M10I1R0\include\Det.h

Files from Rte folder:

- ..\Rte_TS_T40D2M10I1R0\src\Schm_Can.c
- ..\Rte_TS_T40D2M10I1R0\include\Schm_Can.h

Files from EcuM folder:

- ..\EcuM_TS_T40D2M10I1R0\src\EcuM.c
- ..\EcuM_TS_T40D2M10I1R0\include\EcuM.h
- ..\EcuM_TS_T40D2M10I1R0\include\EcuM_Cbk.h

3.3 Setting up the Plugins

The CAN driver was designed to be configured by using the EB Tresos Studio (version EB tresos Studio 24.0.1 b180321-0610 or later.)

Location of various files inside the CAN module folder:

- VSMD (Vendor Specific Module Definition) file in EB tresos Studio XDM format:
 - ..\Can_TS_T40D2M10I1R0\config\Can.xdm
 - ..\CanIf_TS_T40D2M10I1R0\config\CanIf.xdm
 - ..\EcuM_TS_T40D2M10I1R0\config\EcuM.xdm
 - ..\Base_TS_T40D2M10I1R0\config\Base.xdm
 - ..\Resource_TS_T40D2M10I1R0\config\Resource.xdm
 - ..\Mcu_TS_T40D2M10I1R0\config\Mcu.xdm
- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
 - ..\Can_TS_T40D2M10I1R0\autosar\Can.epd
 - ..\CanIf_TS_T40D2M10I1R0\autosar\CanIf.epd
 - ..\EcuM_TS_T40D2M10I1R0\autosar\EcuM.epd
 - ..\Mcu_TS_T40D2M10I1R0\autosar\Mcu.epd
- Code Generation Templates for Pre-Compile time configuration parameters:
 - ..\Can_TS_T40D2M10I1R0\generate_PC\include\Can_Cfg.h
 - ..\Can_TS_T40D2M10I1R0\generate_PC\include\Can_Cfg.c
 - ..\Can_TS_T40D2M10I1R0\generate_PC\Can_SourceClock.m
 - ..\Can_TS_T40D2M10I1R0\generate_PC\Can_VersionCheck_Inc.m
 - ..\Can_TS_T40D2M10I1R0\generate_PC\Can_VersionCheck_Src.m
 - ..\Can_TS_T40D2M10I1R0\generate_PC\Can_NotifyCheck_Src.m
- Code Generation Templates for Post-Build time configuration parameters:
 - ..\Can_TS_T40D2M10I1R0\generate_PB\include\Can_Cfg.h
 - ..\Can_TS_T40D2M10I1R0\generate_PB\src\Can_PBcfg.c
 - ..\Can_TS_T40D2M10I1R0\generate_PB\Can_SourceClock.m
 - ..\Can_TS_T40D2M10I1R0\generate_PB\Can_VersionCheck_Src_PB.m
 - ..\Can_TS_T40D2M10I1R0\generate_PB\Can_NotifyCheck_Src_PB.m

Steps to generate the configuration:

1. Copy the module folders Can_TS_T40D2M10I1R0, CanIf_TS_T40D2M10I1R0, Base_TS_T40D2M10I1R0, Resource_TS_T40D2M10I1R0, EcuM_TS_T40D2M10I1R0, Mcu_TS_T40D2M10I1R0 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 CAN Segment values.
- **RESOURCE** is required to select processor derivative. Current Can driver has support for the following derivatives, everyone having attached a Resource file: s32k148_lqfp144, s32k148_lqfp176, s32k148_mapbga100, s32k146_lqfp144, s32k146_lqfp100, s32k146_lqfp64, s32k146_mapbga100, s32k144_lqfp100, s32k144_lqfp64, s32k144_mapbga100, s32k142_lqfp100, s32k142_lqfp64, s32k118_lqfp48, s32k118_lqfp64, s32k142_lqfp48, s32k144_lqfp48, s32k148_lqfp100.
- **CANIF** is required for reporting status of some events.
- **ECUM** is required for selecting the reference to the wakeup source for every Can controller.
- **DET** is required for signaling the development error detection (parameters out of range, null pointers, etc).

Chapter 4

Function calls to module

4.1 Function Calls during Start-up

The CAN module shall be initialized by Can_Init() service call during the start-up. API service Can_SetController_Mode(Can_Controller, CAN_T_START) shall be used for setting the CAN controller to running mode.

Note

Pin settings are not related to Can driver or plugin configuration. GPIO pins used for connection of CAN physical layer have to be properly assigned to the IPV_FlexCAN module prior the CAN initialization.

4.2 Function Calls during Shutdown

The IPV_FlexCAN IP has many Low Power Modes, with programmable wake up on bus activity.

- **Freeze Mode**

This low power mode is entered when the HALT and FRZ bits in the MCR Register are asserted.

Module ignores the Rx input pin and drives the Tx pin as recessive, stops the prescaler, thus halting all CAN protocol activities and grants write access to the Error Counters Register (ECR), which is read-only in other modes.

Exit from this mode is done by negating the FRZ and HALT bits in the MCR Register or when the MCU is removed from Debug Mode

Note

It is not possible to exit from this mode by receiving a message on the Can bus.

- **Module Disable Mode**

This low power mode is entered when the MDIS bit in the MCR Register is asserted.

Module shuts down the clocks to the CAN Protocol Interface and Message Buffer Management sub-modules.

Exit from this mode is done by negating the MDIS bit in the MCR Register.

Note

It is not possible to exit from this mode by receiving a message on the Can bus.

- **Stop Mode**

This low power mode is entered when Stop Mode is requested at MCU level.

When in Stop Mode, the module puts itself in an inactive state and then informs the CPU that the clocks can be shut down globally.

Exit from this mode happens when the Stop Mode request is removed or when activity is detected on the CAN bus and the Self Wake Up mechanism is enabled.

Note

Note that wake-up from Stop Mode only works when both bits, SLF_WAK and WAK_MSK, are asserted. If interrupt for Wakeup is implemented in INTC, the interrupt handler can change the state of the controller to RUN mode if it is needed. When exit from this mode controller is usually in Freeze mode.

Note

Refer the Reference Manual if wakeup is supported by hardware (bits register are implemented and INTC has connected the CAN wakeup interrupt signal).

4.3 Function Calls during Wake-up

The controller can be wakeup by a message when it is in Sleep mode only if interrupt for wakeup is enabled or self wakeup mechanism is enabled.

CAN stack can be changed from SLEEP mode to STOP mode by calling the `Can_SetControllerMode(CAN_T_WAKEUP)` service call.

Note

Refer the Reference Manual if wakeup is supported by hardware (bits register are implemented and INTC has connected the CAN wakeup interrupt signal).

Chapter 5

Module requirements

5.1 Exclusive areas to be defined in BSW scheduler

CAN_EXCLUSIVE_AREA_00 - Used in “Can_DisableControllerInterrupts” function, to protect the variable for nesting level of enabling/disabling interrupts. Refer to CAN202 requirement.

CAN_EXCLUSIVE_AREA_01 - Used in “Can_EnableControllerInterrupts” function, to protect the variable for nesting level of enabling/disabling interrupts. Refer to CAN202 requirement.

CAN_EXCLUSIVE_AREA_02 - Used in “Can_FlexCan_Write” function to protect the operation for checking the status of MB and for reserving it as a free to use for transmission. If hardware transmit object is free the mutex for that HTH is set to “signaled”. Between this verification and signal operation the protection must be applied. Refer to CAN212 requirement.

CAN_EXCLUSIVE_AREA_03 - Used in “Can_FlexCan_GotoFreezeMode” function to protect the operation which putting controller into Freeze Mode. In this mode, no transmission or reception of frames is done and synchronicity to the CAN bus is lost.

CAN_EXCLUSIVE_AREA_04 - Used in "Can_FlexCan_ResetController" function to protect the operation for enabling the CAN module and requesting a Soft Reset.

CAN_EXCLUSIVE_AREA_05 - Used in "Can_FlexCan_InitController" function, to protect the operation which shall initialize all CAN controllers according to their configuration.

CAN_EXCLUSIVE_AREA_06 - Used in a sub-function of "Can_FlexCan_SetControllerToStopMode" function, to protect the operation to transient all controllers into Stop Mode which puts itself in an inactive state and then informs the CPU that the clocks can be shut down globally.

CAN_EXCLUSIVE_AREA_07 - Used in a sub-function of “Can_FlexCan_DisableInterrupts” function, to protect the operation for disabling interrupts of the Can module.

CAN_EXCLUSIVE_AREA_08 - Used in a sub-function of “Can_FlexCan_EnableInterrupts” function, to protect the operation for enabling interrupts of the Can module.

CAN_EXCLUSIVE_AREA_09 - Used in a sub-function of “Can_FlexCan_SelectClockSource” function, to protect the operation for selecting the clock source to the CAN Protocol Engine (PE) to be either the peripheral clock or the oscillator clock

CAN_EXCLUSIVE_AREA_11 - Used in a sub-function of "Can_FlexCan_InitRxFifo" function to protect the operation for configuring ID Acceptance Mode of the Rx FIFO ID Filter Table elements.

CAN_EXCLUSIVE_AREA_13 - Used in a sub-function of "Can_FlexCan_SetControllerToStartMode" function to protect the operation which transient Controller to Start Mode.

Critical Region Exclusive Matrix

Below is the table depicting the exclusivity between different critical region IDs from the Can driver. If there is an “X” in a table, it means that those 2 critical regions cannot interrupt each other.

The critical regions from interrupts are grouped in “Interrupt Service Routines Critical Regions (composed diagram)”. If an exclusive area is “exclusive” with the composed “Interrupt Service Routines Critical Regions (composed diagram)” group, it means that it is exclusive with each one of the ISR critical regions.

Table 5-1. Exclusive Areas

	CAN_EXCLUSIVE_AREA_00	CAN_EXCLUSIVE_AREA_01	CAN_EXCLUSIVE_AREA_02	CAN_EXCLUSIVE_AREA_03	CAN_EXCLUSIVE_AREA_04	CAN_EXCLUSIVE_AREA_05	CAN_EXCLUSIVE_AREA_06	CAN_EXCLUSIVE_AREA_07	CAN_EXCLUSIVE_AREA_08	CAN_EXCLUSIVE_AREA_09	CAN_EXCLUSIVE_AREA_11	CAN_EXCLUSIVE_AREA_13	Interrupt Service Routines Critical Regions
CAN_EXCLUSIVE_AREA_00		X											

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

	CAN_EXCL USIVE _ARE A_00	CAN_EXCL USIVE _ARE A_01	CAN_EXCL USIVE _ARE A_02	CAN_EXCL USIVE _ARE A_03	CAN_EXCL USIVE _ARE A_04	CAN_EXCL USIVE _ARE A_05	CAN_EXCL USIVE _ARE A_06	CAN_EXCL USIVE _ARE A_07	CAN_EXCL USIVE _ARE A_08	CAN_EXCL USIVE _ARE A_09	CAN_EXCL USIVE _ARE A_11	CAN_EXCL USIVE _ARE A_13	Interru pt Servic e Routin es Critica l Regio ns
CAN_E XCLUS IVE_A REA_0 1	X												
CAN_E XCLUS IVE_A REA_0 2						X							
CAN_E XCLUS IVE_A REA_0 3					X	X	X			X	X	X	
CAN_E XCLUS IVE_A REA_0 4				X		X	X			X	X	X	
CAN_E XCLUS IVE_A REA_0 5			X	X	X		X			X	X	X	
CAN_E XCLUS IVE_A REA_0 6				X	X	X		X	X	X	X	X	
CAN_E XCLUS IVE_A REA_0 7							X		X	X			
CAN_E XCLUS IVE_A REA_0 8							X	X		X			

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

	CAN_EXCL USIVE _ARE A_00	CAN_EXCL USIVE _ARE A_01	CAN_EXCL USIVE _ARE A_02	CAN_EXCL USIVE _ARE A_03	CAN_EXCL USIVE _ARE A_04	CAN_EXCL USIVE _ARE A_05	CAN_EXCL USIVE _ARE A_06	CAN_EXCL USIVE _ARE A_07	CAN_EXCL USIVE _ARE A_08	CAN_EXCL USIVE _ARE A_09	CAN_EXCL USIVE _ARE A_11	CAN_EXCL USIVE _ARE A_13	Interru pt Servic e Routin es Critica l Region s
CAN_E XCLUS IVE_A REA_0 9				X	X	X	X	X	X		X	X	
CAN_E XCLUS IVE_A REA_1 1				X	X	X	X			X		X	
CAN_E XCLUS IVE_A REA_1 3				X	X	X	X			X	X		
Interrup t Service Routine s Critical Region s													

5.2 Peripheral Hardware Requirements

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

There have to be another one CAN node present on the CAN bus in order to get the CAN bus synchronized.

5.3 ISR to configure within OS – dependencies

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.

NXP Semiconductors 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.

5.4 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.

NXP Semiconductors 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.

5.4.1 The following ISR's are used by the CAN driver

Table 5-2. CAN ISRs

ISR Name	Hardware Interrupt Vector
For CanCodeSizeOptimization = STD_OFF	
Can_IsrFCA_BO	78
Can_IsrFCA_ERR	79
Can_IsrFCA_MB_00_15	81
Can_IsrFCA_MB_16_31	82
Can_IsrFCB_BO	85
Can_IsrFCB_ERR	86
Can_IsrFCB_MB_00_15	88
Can_IsrFCB_MB_16_31	89
Can_IsrFCC_BO	92
Can_IsrFCC_ERR	93
Can_IsrFCC_MB_00_15	95
Can_IsrFCC_MB_16_31	96

5.4.2 Macros for Interrupts

General Interrupts for every controller

`CAN_BOISR(FC)` expands `ISR(Can_IsrFC##FC##_BO)` for BusOff event.

`CAN_ERROR_NOTIFICATION_ENABLE == STD_ON`

CAN_ERRISR(FC) expands ISR(Can_IsrFC##FC##_ERR) for error event.

CanCodeSizeOptimalization = STD_ON: All related ISRs are routed to one ISR function.

CAN_MB_UNIISRS(FC) expands ISR(Can_IsrFC##FC##_UNI) for Rx and Tx MBs.

CAN_MB_UNITXISRS(FC) expands ISR(Can_IsrFC##FC##_UNI) for Tx MBs.

CAN_MB_UNIRXISRS(FC) expands ISR(Can_IsrFC##FC##_UNI) for Rx MBs.

CanCodeSizeOptimalization = STD_OFF: All related ISRs have separate ISR functions taking care about the ISR processing.

CAN_RXFIFO_EVENTS(FC) expands ISR(Can_IsrFC##FC##_RxFifoEvents) for all Fifo events.

CAN_MB_ISRS(FC, Name, IdMin, IdMax) expands ISR(Can_IsrFC##FC##_##Name) for a group of Rx or Tx MBs.

CAN_MB_RXISRS(FC, Name, IdMin, IdMax) expands ISR(Can_IsrFC##FC##_##Name) for a group of Rx MBs.

CAN_MB_TXISRS(FC, Name, IdMin, IdMax) expands ISR(Can_IsrFC##FC##_##Name) for a group of Tx MBs.

Note

S32K14X has the IFLAG1[7:4] bits assigned to the same interrupt, then the solution is to use CAN_RXFIFO_EVENTS interrupts macro generation. The CAN_RXFIFO_EVENT_UNIFIED define is generated by Tressos in Can_Cfg.h file and depends by the attribute “Can.CanConfigSet.RxFifoEventsUnified” from Resource properties file.

Interrupt Handlers Example for Can controller A:

ISR(Can_IsrFCA_BO) or ISR(Can_IsrFCA_FrameAv) ISR(Can_IsrFCA_Overf)
ISR(Can_IsrFCA_Warn)

ISR(Can_IsrFCA_ERR)

ISR(Can_IsrFCA_UNI)

ISR(Can_IsrFCA_RxFifoEvents)

ISR(Can_IsrFCA_MB_00_15)

ISR(Can_IsrFCA_MB_16_31)

Interrupt Handlers Example for Can controller B:

ISR(Can_IsrFCB_BO) or ISR(Can_IsrFCB_FrameAv) ISR(Can_IsrFCB_Overf)
ISR(Can_IsrFCB_Warn)

ISR(Can_IsrFCB_ERR)

ISR(Can_IsrFCB_UNI)

ISR(Can_IsrFCB_RxFifoEvents)

ISR(Can_IsrFCB_MB_00_15)

Interrupt Handlers Example for Can controller C:

ISR(Can_IsrFCC_BO) or ISR(Can_IsrFCC_FrameAv) ISR(Can_IsrFCC_Overf)
ISR(Can_IsrFCC_Warn)

ISR(Can_IsrFCC_ERR)

ISR(Can_IsrFCC_UNI)

ISR(Can_IsrFCC_RxFifoEvents)

ISR(Can_IsrFCC_MB_00_15)

5.5 Other AUTOSAR Modules - Dependencies

- **Base:** The Base module contains the common files/definitions needed by all MCAL modules.
- **Mcu:** The Mcu driver provides services for basic microcontroller initialization, power down functionality, reset and microcontroller specific functions required by other MCAL software modules. The clocks need to be initialized prior to using the Can driver. The clock frequency may affect the Can bit rate, the transmitting or receiving a Can frame process.
- **Port:** The Port module is used to configure the port pins with the needed modes, before they are used by the Can module.
- **EcuC :** (only for ASR 4.2) The ECUC module is used for ECU configuration. Can modules need ECUC to retrieve the variant information.
- **Det** The Det module is used for enabling Development error detection. The API function used is Det_ReportError(). The activation / deactivation of Development error detection is configurable using the 'CanDevErrorDetect' configuration parameter.
- **Resource:** Sub-Derivative model is selected from Resource configuration.

- **Rte:** The Rte module is needed for implementing data consistency of exclusive areas that are used by Can module.
- **EcuM:** This module is used for processing the Wakeup notifications of CAN. Whenever the module is in 'Sleep' mode and a wakeup event occurs, it is reported to EcuM through the EcuM_CheckWakeupEvent() API.

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

The Can module can be run in **user mode** if the following steps are performed:

- Enable CanEnabelUserModeSupport from the configuration
- Call the following function as trusted functions:
 - Can_FlexCan_InitController() to configure the CAN_MCR register which is only able to be accessed in the Supervisor mode.
 - Can_FlexCan_SetControllerMode() to configure the CAN_MCR register for setting the Can controller mode.
 - Can_FlexCan_MainFunctionMode() to read the CAN_MCR register for checking the state transition of the CAN controller.
 - Can_FlexCan_DeInitController() to set CAN_MCR[SOFTTRST] bit for the Soft reset controller request
 - Can_FlexCan_SetIcomConfig() to configure the CAN_MCR register for the Icom (Pretended Networking) configuration
 - Can_FlexCan_DeactivateIcom() to configure the CAN_MCR register for deactivate Icom.

Chapter 6

Main API Requirements

6.1 Main Functions Calls within BSW Scheduler

CAN Driver support 4 main functions that can be configured to be scheduled by BSW scheduler:

- FUNC (void, CAN_CODE) Can_MainFunction_Write(void)
- FUNC (void, CAN_CODE) Can_MainFuction_Read(void)
- FUNC (void, CAN_CODE) Can_MainFunction_BusOff(void)
- FUNC (void, CAN_CODE) Can_MainFunction_Mode(void)

These Autosar APIs are scheduled if these 3 events are configured to be in “Polling” mode by the following parameters:

- CanTxProcessing

```
#define CAN_TXPOLL_SUPPORTED (STD_ON)
```

- CanRxProcessing

```
#define CAN_RXPOLL_SUPPORTED (STD_ON)
```

- CanBusoffProcessing

```
#define CAN_BUSOFFPOLL_SUPPORTED (STD_ON)
```

The period for polling is configured by the following 4 parameters:

- CanMainFunctionWritePeriod

```
#define CAN_MAINFUNCTION_PERIOD_WRITE (uint32)0.0010U
```

- CanMainFunctionReadPeriod

```
#define CAN_MAINFUNCTION_PERIOD_READ (uint32)0.0010U
```

- CanMainFunctionBusoffPeriod

```
#define CAN_MAINFUNCTION_PERIOD_BUSOFF (uint32)0.0010U
```

- CanMainFunctionModePeriod

```
#define CAN_MAINFUNCTION_MODE_PERIOD (uint32)0.0010U
```

Note

A configuration for an hardware unit can be possible in such a way that one controller will handle events by interrupts and another by polling method.

6.2 API Requirements

CAN360,CAN361,CAN362,CAN363,CAN228,CAN112,CAN185,CAN186,CAN235,

6.3 Calls to Notification Functions, Callbacks, Callouts

Call-back Notifications

The CAN stack provides the following call-back notifications:

- CanIf_TxConfirmation: This CAN Interface call-back function is called when a CAN message has been transmitted.

```
FUNC (void, CAN_CODE) CanIf_TxConfirmation(PduIdType CanTxPduId)
```

- CanIf_RxIndication: This CAN Interface call-back function is called when valid CAN message is received.

```
FUNC (void, CAN_CODE) CanIf_RxIndication(uint8 Hrh, Can_IdType CanId, uint8 CanDlc,
uint8Ptr CanSduPtr)
```

- CanIf_CancelTxConfirmation: This CAN Interface call-back function is called when the CAN message has been canceled during the transmission.

```
FUNC (void, CAN_CODE) CanIf_CancelTxConfirmation(const Can_PduType * PduInfoPtr)
```

- CanIf_ControllerBusOff: This CAN Interface call-back function is called when the CAN controller reached the bus-off state (see CAN specification for further details).

```
FUNC (void, CAN_CODE) CanIf_ControllerBusOff(uint8 Controller)
```

User Notification

- None

Chapter 7

Memory Allocation

7.1 Sections to be defined in MemMap.h

Table 7-1. Sections to be defined in MemMap.h

Section name	Type of section	Description
CAN_START_SEC_CONFIG_DATA_UNSPECIFIED	Configuration Data	Start of Memory Section for Config Data
CAN_STOP_SEC_CONFIG_DATA_UNSPECIFIED	Configuration Data	End of Memory Section for Config Data
CAN_START_SEC_CODE	Code	Start of memory Section for Code
CAN_STOP_SEC_CODE	Code	End of memory Section for Code
CAN_START_SEC_VAR_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 initialized with values after every reset.
CAN_STOP_SEC_VAR_INIT_UNSPECIFIED	Variables	End of above section.
CAN_START_SEC_VAR_NO_INIT_UNSPECIFIED	Variables	These variables are never cleared and never initialized by start-up code.
CAN_STOP_SEC_VAR_NO_INIT_UNSPECIFIED	Variables	End of above section.
CAN_START_SEC_CONST_UNSPECIFIED	Constant Data	Used for constants
CAN_STOP_SEC_CONST_UNSPECIFIED	Constant Data	End of above section.

7.2 Linker command file

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

Chapter 8

Configuration parameters considerations

Configuration parameter class for Autosar CAN 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
Can	IMPLEMENTATION_CONFIG_VARIANT	Pre Compile parameter for all Variants of Configuration	Pre compile
CanGeneral	CanDevErrorDetection	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanVersionInfoApi	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanIndex	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanMainFunctionBusoffPeriod	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanMultiplexedTransmission	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanMainFunctionModePeriod	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanMaxMessageBuffers	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanTimeoutDuration	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanLPduReceiveCalloutFunction	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanCounterRef	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanAllowLoopAsCycle	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
	CanNumberOfTimeOutLoop	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanCodeSizeOptimization	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanSupportTTCANRef	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanCodeSizeOptimization	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanExtendedIDSupport	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanMBCountExtensionSupport	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanApiEnableMbAbort	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanEnableDualClockMode	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanSetBaudrateApi	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanAbortOnlyOneMB	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanSpecifiedRAMBlockSize	Pre Compile parameter for all Variants of Configuration	Pre compile
	MBDSR	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanPublicIcomSupport	Pre Compile parameter for all Variants of Configuration	Pre compile
CanGeneral CanMainFunctionRWPeriods	CanMainFunctionPeriod	Pre Compile parameter for all Variants of Configuration	Pre compile
CanController	CanHwChannel	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanControllerActivation	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanControllerBaseAddress	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanControllerId	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanRxProcessing	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanTxProcessing	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanBusoffProcessing	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanBccSupport	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanListenOnlyMode	VariantPC or VariantPB	VariantPC or VariantPB
	CanLoopBackMode	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
	CanSoftwareBusOffRecovery	VariantPC or VariantPB	VariantPC or VariantPB
	CanAutoBusOffRecovery	VariantPC or VariantPB	VariantPC or VariantPB
	CanTrippleSamplingEnable	VariantPC or VariantPB	VariantPC or VariantPB
	CanLowestBuffTransmitFirst	VariantPC or VariantPB	VariantPC or VariantPB
	CanLocalPriorityEn	VariantPC or VariantPB	VariantPC or VariantPB
	CanWarningEnable	VariantPC or VariantPB	VariantPC or VariantPB
	CanClockFromBus	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanCpuClockRef	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanCpuClockRef_Alternate	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanControllerRxFifoEnable	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanRxFifoWarningNotification	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanRxFifoOverflowNotification	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanErrorControllerNotifEn	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanErrorControllerNotification	Pre Compile parameter for all Variants of Configuration	Pre compile
	CanControllerDefaultBaudrate	Pre Compile parameter for all Variants of Configuration	Pre compile
CanController \CanControllerBaudRate	CanControllerBaudRate	VariantPC or VariantPB	VariantPC or VariantPB
	CanControllerCheckCanStand	VariantPC or VariantPB	VariantPC or VariantPB
	CanControllerPropSeg	VariantPC or VariantPB	VariantPC or VariantPB
	CanControllerSeg1	VariantPC or VariantPB	VariantPC or VariantPB
	CanControllerSeg2	VariantPC or VariantPB	VariantPC or VariantPB
	CanSyncJumpWidth	VariantPC or VariantPB	VariantPC or VariantPB
	CanAdvancedSetting	VariantPC or VariantPB	VariantPC or VariantPB
	CanBusLength	VariantPC or VariantPB	VariantPC or VariantPB
	CanPropDelayOfTranceiver	VariantPC or VariantPB	VariantPC or VariantPB
	CanControllerPrescaler	VariantPC or VariantPB	VariantPC or VariantPB
	CanControllerPrescaler_Alt	VariantPC or VariantPB	VariantPC or VariantPB
CanController\CanRxFifo	CanControllerIDAcceptanceMode	VariantPC or VariantPB	VariantPC or VariantPB
	CanIDValue0	VariantPC or VariantPB	VariantPC or VariantPB
	CanIDValue1	VariantPC or VariantPB	VariantPC or VariantPB
	CanIDValue2	VariantPC or VariantPB	VariantPC or VariantPB
	CanIDValue3	VariantPC or VariantPB	VariantPC or VariantPB
	CanTableIDType	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
	CanMBFilterMaskValue	VariantPC or VariantPB	VariantPC or VariantPB
	CanRxFifoFiltersNumber	VariantPC or VariantPB	VariantPC or VariantPB
	CanRxFifoGlobalMaskValue	VariantPC or VariantPB	VariantPC or VariantPB
CanController\CanFilterMask	CanFilterMaskValue	VariantPC or VariantPB	VariantPC or VariantPB
CanController \CanHardwareObject	CanHandleType	VariantPC or VariantPB	VariantPC or VariantPB
	CanIdType	VariantPC or VariantPB	VariantPC or VariantPB
	CanIdValue	VariantPC or VariantPB	VariantPC or VariantPB
	CanMBPrio	VariantPC or VariantPB	VariantPC or VariantPB
	CanObjectId	VariantPC or VariantPB	VariantPC or VariantPB
	CanObjectType	VariantPC or VariantPB	VariantPC or VariantPB
	CanControllerRef	VariantPC or VariantPB	VariantPC or VariantPB
	CanFilterMaskRef	VariantPC or VariantPB	VariantPC or VariantPB
	CanMainFunctionRWPeriodRef	VariantPC or VariantPB	VariantPC or VariantPB
CanController \CanTTController	CanTTControllerApplWatchdogLimit	VariantPC or VariantPB	VariantPC or VariantPB
	CanTTControllerCycleCountMax	VariantPC or VariantPB	VariantPC or VariantPB
	CanTTControllerExpectedTxTrigger	VariantPC or VariantPB	VariantPC or VariantPB
	CanTTControllerExternalClockSynchronisation	VariantPC or VariantPB	VariantPC or VariantPB
	CanTTControllerGlobalTimeFiltering	VariantPC or VariantPB	VariantPC or VariantPB
	CanTTControllerInitialRefOffset	VariantPC or VariantPB	VariantPC or VariantPB
	CanTTControllerInterruptEnable	VariantPC or VariantPB	VariantPC or VariantPB
	CanTTControllerLevel2	VariantPC or VariantPB	VariantPC or VariantPB
	CanTTControllerNTUConfig	VariantPC or VariantPB	VariantPC or VariantPB
	CanTTControllerOperationMode	VariantPC or VariantPB	VariantPC or VariantPB
	CanTTControllerSyncDeviation	VariantPC or VariantPB	VariantPC or VariantPB
	CanTTControllerTURRestore	VariantPC or VariantPB	VariantPC or VariantPB
	CanTTControllerTimeMaster	VariantPC or VariantPB	VariantPC or VariantPB
	CanTTControllerTimeMasterPriority	VariantPC or VariantPB	VariantPC or VariantPB
	CanTTControllerTxEnableWindowLength	VariantPC or VariantPB	VariantPC or VariantPB
	CanTTControllerWatchTriggerGapTimeMark	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
	CanTTControllerWatchTriggerTimeMark	VariantPC or VariantPB	VariantPC or VariantPB
	CanTTIRQProcessing	VariantPC or VariantPB	VariantPC or VariantPB

Chapter 9

Integration Steps

This section gives a brief overview of the steps needed for integrating Controller Area Network :

- Generate the required CAN configurations. For more details refer to section [Files Required for the Compilation](#)
- Allocate proper memory sections in CAN_MemMap.h and linker command file. For more details refer to section [Sections to be defined in MemMap.h](#)
- Compile & build the CAN with all the dependent modules. For more details refer to section [Building the Driver](#)



Chapter 10

ISR Reference

10.1 Function Can_IsrFCx_BO

Interrupt routine for BusOff, when x = A, B, C depending on controller's name.

Details:

Interrupt routine for BusOff.

Return: void.

Prototype: void Can_IsrFCx_BO(void);

10.2 Function Can_IsrFCx_ERR

Interrupt routine for Error, when x = A, B, C depending on controller's name.

Details:

Interrupt routine for Error.

Return: void.

Prototype: void Can_IsrFCx_ERR(void);

10.3 Function Can_IsrFCx_MB_00_15

Process all MBs defined for current controller, but between a definite range, when x = A, B, C depending on controller's name

Details:

Process all MBs defined for current controller, but between a definite range.

Return: void.

Prototype: void Can_IsrFCx_MB_00_15(void);

10.4 Function Can_IsrFCx_MB_16_31

Process all MBs defined for current controller, but between a definite range, when x = A, B, C depending on controller's name

Details:

Process all MBs defined for current controller, but between a definite range.

Return: void.

Prototype: void Can_IsrFCx_MB_16_31(void);

10.5 Function Can_IsrFCx_RxFifoEventsMbMix

Interrupt routine for Legacy Rx fifo Support, when x = A, B, C depending on controller's name.

Details:

Interrupt routine for Legacy Rx fifo Support.

Return: void.

Prototype: void Can_IsrFCx_RxFifoEventsMbMix(void);

10.6 Function Can_IsrFCx_UNI

Interrupt routine for all MBs - ISR optimization code size, when x = A, B, C depending on controller's name.

Details:

Interrupt routine for all MBs - ISR optimization code size.

Return: void.

Prototype: `void Can_IsrFCx_UNI(void);`

Chapter 11

External Assumptions for CAN driver

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

[SMCAL_CPR_EXT4]

<< The external application shall call Can_Init function only when the driver state is CAN_UNINIT and the state of all controllers is UNINIT. >>

[SMCAL_CPR_EXT5]

<< The external application shall call Can_MainFunction_Write only after driver initialization. >>

[SMCAL_CPR_EXT6]

<< The external application shall call Can_MainFunction_Read only after driver initialization. >>

[SMCAL_CPR_EXT7]

<< The external application shall call Can_MainFunction_BusOff only after driver initialization. >>

[SMCAL_CPR_EXT9]

<< The external application shall call Can_SetControllerMode only after driver initialization >>

[SMCAL_CPR_EXT10]

<< The external application shall call Can_DisableControllerInterrupts function only after driver initialization. >>

[SMCAL_CPR_EXT11]

<< The external application shall call Can_EnableControllerInterrupts function only after driver initialization. >>

[SMCAL_CPR_EXT12]

<< The external application shall call Can_Write function only after driver initialization. >>

[SMCAL_CPR_EXT13]

<< The external application shall assure that Can_Init does not preempt and is not preempted by any other CAN driver API excepting Can_GetVersionInfo.

The external application shall assure that Can_Init does not preempt itself. >>

[SMCAL_CPR_EXT14]

<< The external application shall assure that Can_MainFunction_Write does not preempt and is not preempted by any other CAN driver API exceptin Can_GetVersionInfo.The external application shall assure that Can_MainFunction_Write does not preempt itself. >>

[SMCAL_CPR_EXT15]

<< The external application shall assure that Can_MainFunction_Read does not preempt and is not preempted by any other CAN driver API exceptin Can_GetVersionInfo.The external application shall assure that Can_MainFunction_Read does not preempt itself. >>

[SMCAL_CPR_EXT16]

<< The external application shall assure that Can_MainFunction_BusOff does not preempt and is not preempted by any other CAN driver API exceptin Can_GetVersionInfo.The external application shall assure that Can_MainFunction_BusOff does not preempt itself. >>

[SMCAL_CPR_EXT17]

<< The external application shall assure that Can_SetControllerMode does not preempt and is not preempted by any other CAN driver API using the same controller parameter.

The external application shall assure that Can_SetControllerMode does not preempt itself. >>

[SMCAL_CPR_EXT18]

<< The external application shall assure that Can_DisableControllerInterrupts does not preempt and is not preempted by any other CAN driver API using the same controller parameter. >>

[SMCAL_CPR_EXT19]

<< The external application shall assure that Can_EnableControllerInterrupts does not preempt and is not preempted by any other CAN driver API using the same controller parameter. >>

[SMCAL_CPR_EXT20]

<< The external application shall assure that Can_Write does not preempt and is not preempted by any other CAN driver API using the same controller as the hardware handle parameter. The external application shall assure that Can_Write does not preempt itself for the same hardware handle parameter. >>

[SMCAL_CPR_EXT22]

<< The external application shall call Can_Init function only when the driver state is CAN_UNINIT and the state of all controllers is UNINIT. >>

[SMCAL_CPR_EXT25]

<< The external application shall call Can_MainFunction_Mode function only after driver initialization. >>

[SMCAL_CPR_EXT26]

<< The external application shall call Can_Init function only when the driver state is CAN_UNINIT and the state of all controllers is UNINIT. >>

[SMCAL_CPR_EXT27]

<< The external application shall call Can_SetControllerMode(CAN_T_START) only when the CAN controller is in state STOPPED. >>

[SMCAL_CPR_EXT28]

<< The external application shall call Can_SetControllerMode(CAN_T_STOP) only when the CAN controller is in state STARTED or STOPPED. >>

[SMCAL_CPR_EXT29]

<< The external application shall call Can_SetControllerMode(CAN_T_SLEEP) only when the CAN controller is in state SLEEP or STOPPED. >>

[SMCAL_CPR_EXT163]

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

[CAN023]

<< The consistency of the configuration must be checked by the configuration tool(s). >>

[CAN024]

<< The valid values that can be configured are hardware dependent. Therefore the rules and constraints can't be given in the standard. The configuration tool is responsible to do a static configuration checking, also regarding dependencies between modules (i.e. Port driver, MCU driver etc.) >>

[CAN039]

<< Can_ReturnType

Return values of CAN driver API

CAN_OK

CAN_NOT_OK

CAN_BUSY >>

NOTE

defined into Can_GeneralTypes.h included into Base module

[CAN240]

<< The Mcu module (SPAL see [8]) shall configure register settings that are 'shared' with other modules. >>

NOTE

not a requirement for CAN module

[CAN285]

<< Transmit cancellation may only be used when transmit buffers are enabled inside the CanIf module. >>

NOTE

This is a specification for CanIf module.Safety manual

[CAN415]

<< Can_PduType

This type is used to provide ID, DLC and SDU from CAN interface to CAN driver. >>

NOTE

defined into Can_GeneralTypes.h included into Base module

[CAN416]

<< Can_IdType

Represents the Identifier of an L-PDU. For extended IDs the most significant bit is set. >>

NOTE

defined into Can_GeneralTypes.h included into Base module

[CAN417]

<< Can_StateTransitionType

CAN_T_START: CAN controller transition value to request state STARTED.

CAN_T_STOP: CAN controller transition value to request state STOPPED.

CAN_T_SLEEP: CAN controller transition value to request state SLEEP.

CAN_T_WAKEUP: CAN controller transition value to request state STOPPED from state SLEEP.

State transitions that are used by the function CAN_SetControllerMode >>

NOTE

defined into Can_GeneralTypes.h included into Base module

[CAN429]

<< Can_HwHandleType

Represents the hardware object handles of a CAN hardware unit. For CAN hardware units with more than 255 HW objects use extended range. >>

NOTE

defined into Can_GeneralTypes.h included into Base module

[CAN436]

<< Can_GeneralTypes.h shall contain all types and constants that are shared among the AUTOSAR CAN modules Can, CanIf and CanTrcv. >>

NOTE

Implemented in base

[CAN437]

<< The integrator of the Can modules shall provide the file Can_GeneralTypes.h. >>

[CAN438]

<< The content of Can_GeneralTypes.h consists of types specified within [5] and the following type specifications within this document except Can_ConfigType. >>

NOTE

Implemented in base

How to Reach Us:**Home Page:**nxp.com**Web Support:**nxp.com/support

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

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

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

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, μ Vision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2019 NXP B.V.

Document Number IM2CANASR4.3 Rev0001R1.0.1
Revision 1.0