Integration Manual

for S32K14X I2C Driver

Contents

Section number	Title	Page

Chapter 1
Revision History

Chapter 2
Introduction

Chapter 3
Building the Driver

Chapter 4
Function calls to module

Chapter 5
Module requirements

Integration Manual, Rev. 1.0

Chapter 6
Main API Requirements

Chapter 7
Memory Allocation

Chapter 8
Configuration parameters considerations

Chapter 9
Integration Steps

Chapter 10
ISR Reference

Chapter 11
External Assumptions for I2C 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 I2c Driver for S32K14X microcontrollers.

2.1 Supported Derivatives

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

Table 2-1. S32K14X Derivatives

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

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

2.2 Overview

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

AUTOSAR

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

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

2.3 About this Manual

This Technical Reference employs the following typographical conventions:

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

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

Notes and warnings are shown as below:

Note

This is a note.

2.4 Acronyms and Definitions

Table 2-2.   Acronyms and Definitions

Term	Definition
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
ASM	Assembler
BSW	Basic Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
C/CPP	C and C++ Source Code
ECU	Electronic Control Unit
I2C	Inter-Integrated Circuit
ISR	Interrupt Service Routine
N/A	Not Applicable
VLE	Variable Length Encoding

Integration Manual, Rev. 1.0

2.5   Reference List

Table 2-3.   Reference List

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

2.5.1   Reference for I2C Bus Specification

Table 2-4.   I2C Bus Specification and user manual

#	Title	Version
1	I2C Bus Specification and user manual	http://www.nxp.com/documents/user_manual/ UM10204.pdf

Chapter 3
Building the Driver

This section describes the source files and various compilers, linker options used for building the Autosar I2c driver for NXP SemiconductorsS32K14X . It also explains the EB Tresos Studio plugin setup procedure.

3.1 Build Options

The I2c 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   GCC Compiler/Linker/Assembler Options

Table 3-4. Compiler Options

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

Table continues on the next page...

Integration Manual, Rev. 1.0

Table 3-4. Compiler Options (continued)

Option	Description
-fno-common	Specifies that the compiler should place uninitialized global variables in the data section of the object file, rather than generating them as common blocks.
-Wall	Enables all the warnings about constructions that some users consider questionable, and that are easy to avoid even in conjunction with macros.
-Wextra	Enables some extra warning flags that are not enabled by '-Wall'.
-Wstrict-prototypes	Warn if a function is declared or defined without specifying the argument types.
-Wno-sign-compare	Do not warn when a comparison between signed and unsigned values could produce an incorrect result when the signed value is converted to unsigned.
-fstack-usage	Geneates an extra file that specifies the maximum amount of stack used, on a per-function basis.
-fdump-ipa-all	Enables all inter-procedural analysis dumps.
-Werror=implicit-function-declaration	Generates an error when the prototype of the function is not defined..
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DGCC	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the GCC preprocessor symbol.
-std=c99	C programming language standard version c99

Table 3-5. Assembler Options

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

Table 3-6. Linker Options

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

Table continues on the next page...

Integration Manual, Rev. 1.0

Table 3-6.   Linker Options (continued)

Option	Description
-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

3.1.3   IAR Compiler/Linker/Assembler Options

Table 3-7.   Compiler Options

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

Integration Manual, Rev. 1.0

Table 3-8. Assembler Options

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

Table 3-9. Linker Options

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

3.2 Files required for Compilation

This section describes the include files required to compile, assemble (if assembler code) and link the I2c driver for S32K14X microcontrollers.

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

I2c Files
- I2c_TS_T40D2M10I1R0\src\I2c.c
- I2c_TS_T40D2M10I1R0\src\I2c_Ipw.c
- I2c_TS_T40D2M10I1R0\include\I2c.h
- I2c_TS_T40D2M10I1R0\include\I2c_Types.h
- I2c_TS_T40D2M10I1R0\include\I2c_IPW.h
- I2c_TS_T40D2M10I1R0\include\I2c_IPW_Types.h
- I2c_TS_T40D2M10I1R0\src\I2c_LPI2C.c
- I2c_TS_T40D2M10I1R0\src\I2c_FlexIO.c
- I2c_TS_T40D2M10I1R0\src\I2c_LPI2C_Irq.c
- I2c_TS_T40D2M10I1R0\src\I2c_FlexIO_Irq.c

- I2c_TS_T40D2M10I1R0\include\I2c_LPI2C.h
- I2c_TS_T40D2M10I1R0\include\I2c_LPI2C_Types.h
- I2c_TS_T40D2M10I1R0\include\I2c_FlexIO_Types.h
- I2c_TS_T40D2M10I1R0\include\Reg_eSys_LPI2C.h

I2c Generated Files
- I2c_Cfg.c - This file should be generated by the user using a configuration tool for compilation
- I2c_Cfg.h - This file should be generated by the user using a configuration tool for compilation
- I2c_[VariantName]_PBcfg.c - This file should be generated by the user using a configuration tool for compilation

Note

As a deviation from standard:

- I2c_[VariantName]_PBcfg.c - This file will contain the definition for all parameters that are variant aware, independent of the configuration class that will be selected (PC, LT, PB)
- I2c_Cfg.c - This 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 I2c_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_Cfg.h
- Base_TS_T40D2M10I1R0\include\ComStack_Types.h
- Base_TS_T40D2M10I1R0\include\Mcal.h
- Base_TS_T40D2M10I1R0\include\I2c_MemMap.h
- Base_TS_T40D2M10I1R0\include\Platform_Types.h
- Base_TS_T40D2M10I1R0\include\Reg_eSys.h
- Base_TS_T40D2M10I1R0\include\Reg_Macros.h
- Base_TS_T40D2M10I1R0\include\Soc_Ips.h
- Base_TS_T40D2M10I1R0\include\Std_Types.h
- Base_TS_T40D2M10I1R0\generate_PC\include\modules.h

Files from Dem folder:
- Dem_TS_T40D2M10I1R0\include\Dem.h
- Dem_TS_T40D2M10I1R0\include\Dem_Types.h
- Dem_TS_T40D2M10I1R0\generate_PC\include\Dem_IntErrId.h
- Dem_TS_T40D2M10I1R0\src\Dem.c

Files from Det folder:
- Det_TS_T40D2M10I1R0\include\Det.h
- Det_TS_T40D2M10I1R0\src\Det.c

Files from Mcl folder (only when DMA is used):
- Mcl_TS_T40D2M10I1R0\include\CDD_Mcl.h

Files from Rte folder:
- Rte_TS_T40D2M10I1R0\include\SchM_I2c.h
- Rte_TS_T40D2M10I1R0\src\SchM_I2c.c

3.3 Setting up the Plug-ins

The I2c 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 I2c module folder:
- VSMD (Vendor Specific Module Definition) file in EB tresos Studio XDM format:
  - ..\ I2c _ TS_T40D2M10I1R0 \config\I2c.xdm

- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - ..\ I2c _ TS_T40D2M10I1R0 \autosar\I2c_<subderivative_name>.epd

- Code Generation Templates for variant aware parameters:
  - ..\ I2c _ TS_T40D2M10I1R0 \generate_PB\src\I2c_PBcfg.c

- Code Generation Templates for parameters without variation points
  - ..\ I2c _ TS_T40D2M10I1R0 \generate_PC\src\I2c_Cfg.c
  - ..\ I2c _ TS_T40D2M10I1R0 \generate_PC\include\I2c_Cfg.h

Steps to generate the configuration:
1. Copy the module folders I2C _ TS_T40D2M10I1R0 , Base_ TS_T40D2M10I1R0 , Dem_ TS_T40D2M10I1R0 , Det_ TS_T40D2M10I1R0 , EcuC_ TS_T40D2M10I1R0 , Rte_ TS_T40D2M10I1R0 , Resource_ TS_T40D2M10I1R0 , Mcu_ TS_T40D2M10I1R0 , Mcl_ 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

- **RESOURCE** is required to select processor derivative. The I2c 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 .
- **MCU** is required for selecting the clock reference for calculating the baud rate.
- **DET** is required for signalling the development error detection (parameters out of range, null pointers, etc).
- **DEM** is required for signalling the production error detection (hardware failure, etc).
- **MCL** is required when DMA is used for configuring the DMA channels.
- **ECUC** is required for configuring post build variants.

3.3.1 DMA configuration

This section applies only to master channels configured for using DMA as the method used for asynchronous transmissions.

To configure an I2C master channel to use DMA for asynchronous transmission the following steps must be followed:

1) The Mcl driver must be used as a dependency and configured with Mcl DMA Supported (EnableDMA) set to true and the Mcl interrupts enabled in Mcl Interrupts Available tab.
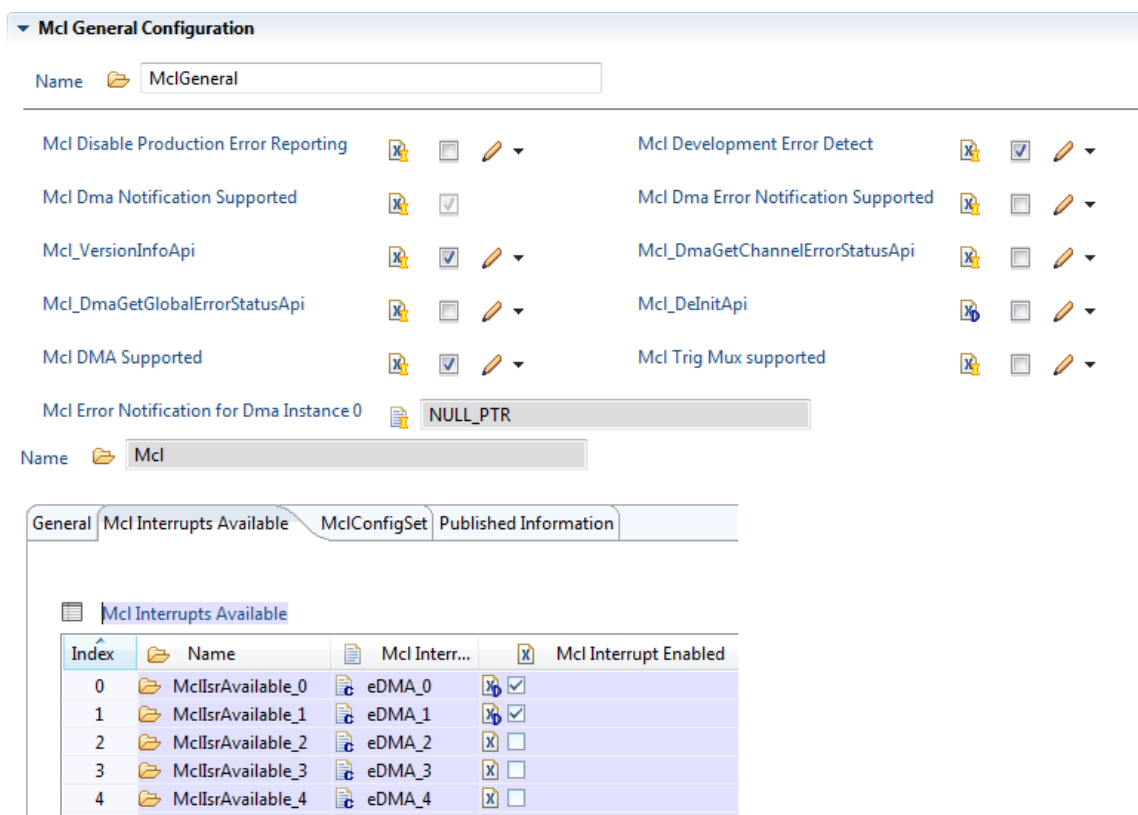
Figure 3-1. Mcl driver configuration options required

2) a) A DMA channel must be configured in the Mcl driver having the following properties:

a. DMA Channel Enable (MclDMAChannelEnable) set to true

b. The DMA Source (DmaSource0) set to LPI2C0_TX, LPI2C1_TX, FLEXIO_SHIFTER0 (for a FLEXIO_0_CH_0_1 channel) or FLEXIO_SHIFTER2 (for a FLEXIO_0_CH_2_3 channel)

c. Mcl Dma Transfer Completion User Notification (MclDmaTransferCompletionNotif) set with a fixed string:

I2c_LPI2C0_DmaTxCompleteNotification for a LPI2C0 channel

I2c_LPI2C1_DmaTxCompleteNotification for a LPI2C1 channel

I2c_FlexIO0_DmaTransferCompleteNotificationShifter0 for a FLEXIO_0_CH_0_1 channel

I2c_FlexIO0_DmaTransferCompleteNotificationShifter2 for a FLEXIO_0_CH_2_3 channel

2) b) A DMA channel must be configured in the Mcl driver having the following properties:

Integration Manual, Rev. 1.0

a. DMA Channel Enable (MclDMAChannelEnable) set to true

b. The DMA Source (DmaSource0) set to LPI2C0_RX, LPI2C1_RX, FLEXIO_SHIFTER1 (for a FLEXIO_0_CH_0_1 channel) or FLEXIO_SHIFTER3 (for a FLEXIO_0_CH_2_3 channel)

c. Mcl Dma Transfer Completion User Notification (MclDmaTransferCompletionNotif) set with a fixed string:

I2c_LPI2C0_DmaRxCompleteNotification for a LPI2C0 channel

I2c_LPI2C1_DmaRxCompleteNotification for a LPI2C1 channel

I2c_FlexIO0_DmaTransferCompleteNotificationShifter1 for a FLEXIO_0_CH_0_1 channel

I2c_FlexIO0_DmaTransferCompleteNotificationShifter3 for a FLEXIO_0_CH_2_3 channel

Integration Manual, Rev. 1.0

Name   DMAChannel_0

General

DMA Channel ID (0 -> 15)   0

DmaHwChannel   eDMA_0

DMA Channel Priority (0 -> 15)   0

ECP   DPA

EMI

Mcl Dma Transfer Completion User Notification   I2C_LPI2C0_DmaTxCompleteNotification

DMA Channel Enable ☑   DMA Channel Trigger Enable

DMA Source 0   LPI2C0_TX

Name   DMAChannel_1

General

DMA Channel ID (0 -> 15)   1

DmaHwChannel   eDMA_1

DMA Channel Priority (0 -> 15)   1

ECP   DPA

EMI

Mcl Dma Transfer Completion User Notification   I2C_LPI2C0_DmaRxCompleteNotification

DMA Channel Enable ☑   DMA Channel Trigger Enable

DMA Source 0   LPI2C0_RX

Figure 3-2. Example of configuring a DMA channel used for a LPI2C0 master channel

Figure 3-3. Example of configuring a DMA channel used for a FLEXIO_0_CH_2_3 master channel

3) In the I2C configuration an I2C channel must be configured as master (I2cMasterSlaveConfiguration is MASTER_MODE), using DMA as the asynchronous method for transmissions (I2cAsyncMethod is DMA).

Each LPI2C unit configured in DMA mode require one DMA channel used for sending and one DMA channel for receiving data:

Integration Manual, Rev. 1.0

I2C Channel ID (0 -> 4)	0
I2C hardware channel	LPI2C_0
I2C master/slave configuration	MASTER_MODE
I2C pin configuration	PINCFG_2PIN_PUSH_PULL

▼ LPI2C Master configuration

Name   I2CMasterConfiguration

I2C Master enabled in debug mode	☑	I2C Master enabled in Doze mode	☑
I2CClockRef	/Mcu/Mcu/McuModuleConfiguration_0/McuClockSettingConfig_0/SLOW_SOSC
I2C Asynchronous Method	INTERRUPT
I2C TX DMA Channel	/Mcl/Mcl/MclConfigSet_0/DMAChannel_0
I2C RX DMA Channel	/Mcl/Mcl/MclConfigSet_0/DMAChannel_1
I2C Prescaler	DIVIDE_BY_1
I2C Glitch Filter SDA (cycles) (0 -> 15)	0
I2C Glitch Filter SCL (cycles) (0 -> 15)	0
I2C Bus Idle Timeout (cycles) (0 -> 4096)	0
I2C Pin Low Timeout (cycles) (0 -> 4095)	0
I2C Data Valid Delay (cycles) (1 -> 63)	2

Figure 3-4. Example for a LPI2C master channel configuration using DMA as asynchronous transfer method

Integration Manual, Rev. 1.0

Figure 3-5. Example for a FlexIO master channel configuration using DMA as asynchronous transfer method

4) The Mcl driver must be initialized prior to calling I2c_AsyncTransmit with DMA asynchronous method and the DMA interrupt handlers must be mapped according to the Mcl Integration Manual .

Integration Manual, Rev. 1.0

Chapter 4
Function calls to module

4.1 Function Calls during Start-up

I2c shall be initialized during STARTUP phase of EcuM initialization. The API to be called for this is I2C _Init(). The MCU module should be initialized before the I2c is initialized. If DMA is used the MCL driver should be initialized before I2c_AsyncTransmit is called for the first time.

4.2 Function Calls during Shutdown

NA.

4.3 Function Calls during Wake-up

NA.

Chapter 5
Module requirements

5.1 Exclusive areas to be defined in BSW scheduler

I2C_EXCLUSIVE_AREA_00 is used to protect I2c_aeChannelStatus in I2c_SyncTransmit from itself and I2c_AsyncTransmit so that a new data transfer can't start while another transfer is being initiated.

I2C_EXCLUSIVE_AREA_01 is used to protect I2c_aeChannelStatus in I2c_AsyncTransmit from itself and I2c_SyncTransmit so that a new data transfer can't start while another transfer is being initiated.

I2C_EXCLUSIVE_AREA_02 is used to protect I2c_aeChannelStatus in I2c_StartListening from itself and I2c_LPI2C_SlaveInterruptProcessing so that it won't affect the integrity of the channel status.

I2C_EXCLUSIVE_AREA_03 is used to protect I2c_aeChannelStatus in I2c_LPI2C_SlaveInterruptProcessing from I2c_StartListening so that it won't affect the integrity of the channel status.

Critical Region Exclusive Matrix

Below is the table depicting the exclusivity between different critical region IDs from the I2c driver. If there is an "X" in a table, it means that those 2 critical regions cannot interrupt each other.

Table 5-1. Exclusive Areas

	I2C_EXCLUSIVE_AREA_00	I2C_EXCLUSIVE_AREA_01	I2C_EXCLUSIVE_AREA_02	I2C_EXCLUSIVE_AREA_03	I2C_EXCLUSIVE_AREA_04	I2C_EXCLUSIVE_AREA_05
I2C_EXCLUSIVE_AREA_00	X	X				
I2C_EXCLUSIVE_AREA_01	X	X				

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

	I2C_EXCLUSIVE_AREA_00	I2C_EXCLUSIVE_AREA_01	I2C_EXCLUSIVE_AREA_02	I2C_EXCLUSIVE_AREA_03	I2C_EXCLUSIVE_AREA_04	I2C_EXCLUSIVE_AREA_05
I2C_EXCLUSIVE_AREA_02			X	X		
I2C_EXCLUSIVE_AREA_03			X	X		

5.2 Peripheral Hardware Requirements

None.

5.3 ISR to configure within OS – dependencies

Table 5-2. I2C ISRs for S32K14X

ISR Name	Hardware interrupt vector
I2c_LPI2C_0_MasterIsr	24
I2c_LPI2C_0_SlaveIsr	25
I2c_LPI2C_1_MasterIsr	29
I2c_LPI2C_1_SlaveIsr	30
Mcl_FlexioProcessCombinedInterrupt	69

Table 5-3. I2C ISRs for S32K118

ISR Name	Hardware interrupt vector
I2c_LPI2C_Isr_LPI2C_0	24
Mcl_FlexioProcessCombinedInterrupt	25

5.4 ISR Macro

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

a. OS is not used - AUTOSAR_OS_NOT_USED is defined:

i. If USE_SW_VECTOR_MODE is defined:

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

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

ii. If USE_SW_VECTOR_MODE is not defined:

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

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

Custom OS is used - AUTOSAR_OS_NOT_USED is not defined

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

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

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

5.5   Other AUTOSAR modules - dependencies
- **Base:** The BASE module contains the common files/definitions needed by all MCAL modules.
- **Mcu:** The MCU driver provides services for basic microcontroller initialization, power down functionality, reset and microcontroller specific functions required by other MCAL software modules. The clocks need to be initialized prior to using the I2C driver. The I2C reference clock is provided by MCU plugin.
- **Port:** The PORT module is used to configure the port pins with the needed modes, before they are used by the I2C module. For each channel, the SCL and SDA signals need to be configured. Please refer to the chapter "Hardware Resources" in User Manual file.
- **Mcl:** For each I2C channel in use, a transmit and a receive DMA channel need to be defined and routed through the DMA Multiplexer using MCL plugin. MCL should be initialized before I2C switch to DMA mode.
- **EcuC:** The ECUC module is used for ECU configuration. MCAL modules need ECUC to retrieve the variant information.
- **Det:** The DET module is used for enabling Default Error Tracing. The API function used is Det_ReportError(). The activation/deactivation of Default Error Tracing is configurable using the 'I2cDevErrorDetect' configuration parameter.
- **Dem:** The DEM module is used for enabling reporting of production relevant error status. The API function used is Dem_SetEventStatus(). The activation/deactivation

Integration Manual, Rev. 1.0

of DEM is configurable using the 'I2cDisableDemReportErrorStatus' configuration parameter.
- **Resource:** The RESOURCE module is used to select microcontroller's derivatives.
- **RTE:** The RTE module is needed for implementing data consistency of exclusive areas that are used by I2C module.

5.6 Data Cache Restriction

In the DMA transfer mode, DMA transfers may issue cache coherency problems. To avoid possible coherency issues when **D-CACHE** is enabled, the user shall ensure that the buffers used as TCD source and destination are allocated in the **NON-CACHEABLE** area (by means of [I2c_]Memmap). Otherwise, the I2C driver has some dependencies. The user must follow the below things:

- User must to put all variables, which were used for transmitter and receiver, to the **NON CACHEABLE** memory section in the RAM zone by the definition **I2C_START_SEC_VAR_<INIT_POLICY>_<ALIGNMENT>_NO_CACHEABLE** and **I2C_STOP_SEC_VAR_<INIT_POLICY>_<ALIGNMENT>_NO_CACHEABLE**

5.7 User Mode Support

I2C module does not include registers protection. So, it is accessible to all registered in any public mode.

Chapter 6
Main API Requirements

6.1 Main functions calls within BSW scheduler

None.

6.2 API Requirements

None.

6.3 Calls to Notification Functions, Callbacks, Callouts

Configurable User Notifications

The I2C driver provides the following configurable user notifications:

- `I2C_ERROR_NOTIFICATION`: This will be called from the I2c_LPI2C_ErrorHandler using I2C_ERROR_NOTIFICATION(u8Channel, u8ErrorCode). It will have two uint8 parameters: the first will be the logical I2C channel, and the second the error code depending.

- `I2C_SLAVE_ADDR_MATCH_NOTIFICATION`: This is used only by slave channels (only LPI2C supports slave channels). This will be called from the I2c_LPI2C_SlaveInterruptProcessing using I2C_SLAVE_ADDR_MATCH_NOTIFICATION(u8Channel, eDirection) when the slave is addressed by a master channel. It will have two uint8 parameters: the first will be the logical I2C channel, and the second the direction of the data request from the master. In this callback the API I2c_PrepareSlaveBuffer should be called to prepare the data buffer used in the communication with the master channel.

- `I2C_SLAVE_TRANSMIT_COMPLETE_NOTIFICATION`: This is used only by slave channels. This will be called from the I2c_LPI2C_SlaveInterruptProcessing using I2C_SLAVE_TRANSMIT_COMPLETE_NOTIFICATION(u8Channel, u8NumberOfBytes) when the slave transmission is finished. It will have two uint8 parameters: the first will be the I2C channel, and the second the number of bytes transmitted.

- `I2C_SLAVE_RECEIVE_COMPLETE_NOTIFICATION`: This is used only by slave channels. This will be called from the I2c_LPI2C_SlaveReceive using I2C_SLAVE_RECEIVE_COMPLETE_NOTIFICATION(u8Channel, u8NumberOfBytes) when the slave reception is finished. It will have two uint8 parameters: the first will be the I2C channel, and the second the number of bytes received.

- `I2C_MASTER_TRANSMIT_COMPLETE_NOTIFICATION`: This is used only by master channels. This will be called from the I2c_LPI2C_MasterInterruptProcessing using I2C_MASTER_TRANSMIT_COMPLETE_NOTIFICATION(u8Channel, u8NumberOfBytes) when the master transmission is finished. It will have two uint8 parameters: the first will be the I2C channel, and the second the number of bytes transmitted.

- `I2C_MASTER_RECEIVE_COMPLETE_NOTIFICATION`: This is used only by master channels. This will be called from the I2c_LPI2C_MasterInterruptProcessing using I2C_MASTER_RECEIVE_COMPLETE_NOTIFICATION(u8Channel, u8NumberOfBytes) when the master reception is finished. It will have two uint8 parameters: the first will be the I2C channel, and the second the number of bytes received.

Chapter 7
Memory Allocation

7.1   Sections to be defined in [I2c_]MemMap.h

Table 7-1.   Memory Allocation

Section name	Type of section	Description
I2C_START_SEC_CONFIG_DATA_UNSPECIFIED	Configuration Data	Start of Memory Section for Config Data
I2C_STOP_SEC_CONFIG_DATA_UNSPECIFIED	Configuration Data	End of Memory Section for Config Data
I2C_START_SEC_CODE	Code	Start of memory Section for Code
I2C_STOP_SEC_CODE	Code	End of memory Section for Code
I2C_START_SEC_RAMCODE	Code	Start of memory Section for Code to be located in RAM
I2C_STOP_SEC_RAMCODE	Code	End of memory Section for Code to be located in RAM
I2C_START_SEC_VAR_NO_INIT_UNSPECIFIED	Variables	Used for variables, structures, arrays when the SIZE (alignment) does not fit the criteria of 8, 16 or 32 bit. These variables are never cleared and never initialized by start-up code.
I2C_STOP_SEC_VAR_NO_INIT_UNSPECIFIED	Variables	End of above section.
I2C_START_SEC_VAR_INIT_32	Variables	Used for variables which have to be aligned to 32 bit. For instance used for variables of size 32 bit or used for composite data types: arrays , structs containing elements of maximum 32 bits. These variables are initialized with values after every reset.
I2C_STOP_SEC_VAR_INIT_32	Variables	End of above section.
I2C_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.
I2C_STOP_SEC_VAR_INIT_UNSPECIFIED	Variables	End of above section.

Table continues on the next page...

Integration Manual, Rev. 1.0

Table 7-1. Memory Allocation (continued)

Section name	Type of section	Description
I2C_START_SEC_CONST_32	Constant Data	Used for constants that have to be aligned to 32 bit.
I2C_STOP_SEC_CONST_32	Constant Data	End of above section.

7.2 Linker command file

Memory shall be allocated for every section defined in I2c_MemMap.h

Chapter 8
Configuration parameters considerations

Configuration parameter class for Autosar I2c 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
I2c	IMPLEMENTATION_CONFIG_VARIANT	Pre Compile parameter for all Variants of Configuration	Pre Compile
GeneralConfiguration	I2cDevErrorDetect	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cDisableDemReportErrorStatus	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cDmaUsed	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cFlexIOUsed	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cTimeoutDuration	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cVersionInfoApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cErrorNotification	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cMasterTransmitCompleteNotification	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cMasterReceiveCompleteNotification	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cSlaveAddressMatchNotification	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cSlaveTransmitCompleteNotification	Pre Compile parameter for all Variants of Configuration	Pre Compile

Table continues on the next page...

Integration Manual, Rev. 1.0

Table 8-1.   Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
	I2cSlaveReceiveCompleteNotification	Pre Compile parameter for all Variants of Configuration	Pre Compile
I2cEnableUserModeSupport	Pre Compile parameter for all Variants of Configuration	Pre Compile	
I2cGlobalConfig/ I2cFlexIOModuleConfiguration	I2cFlexIOEnabledInDebug	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cFlexIOEnabledInDozeMode	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cFlexIOFastAccessMode	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cClockRef	Pre Compile parameter for all Variants of Configuration	Pre Compile
I2cGlobalConfig/I2cChannel	I2cChannelId	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cHwChannel	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cMasterSlaveConfiguration	VariantPC or VariantPB	VariantPC or VariantPB
	I2cPinConfiguration	Pre Compile parameter for all Variants of Configuration	Pre Compile
I2cGlobalConfig/I2cChannel/ I2cMasterConfiguration	I2cMasterEnabledInDebug	VariantPC or VariantPB	VariantPC or VariantPB
	I2cMasterEnabledInDozeMode	VariantPC or VariantPB	VariantPC or VariantPB
	I2cClockRef	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cAsyncMethod	VariantPC or VariantPB	VariantPC or VariantPB
	I2cDmaTxChannelRef	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cDmaRxChannelRef	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cPrescaler	VariantPC or VariantPB	VariantPC or VariantPB
	I2cGlitchFilterSDA	VariantPC or VariantPB	VariantPC or VariantPB
	I2cGlitchFilterSCL	VariantPC or VariantPB	VariantPC or VariantPB
	I2cBusIdleTimeout	VariantPC or VariantPB	VariantPC or VariantPB
	I2cPinLowTimeout	VariantPC or VariantPB	VariantPC or VariantPB
	I2cDataValidDelay	VariantPC or VariantPB	VariantPC or VariantPB
	I2cSetupHoldDelay	VariantPC or VariantPB	VariantPC or VariantPB
	I2cClockHighPeriod	VariantPC or VariantPB	VariantPC or VariantPB
	I2cClockLowPeriod	VariantPC or VariantPB	VariantPC or VariantPB
	I2cBaudRate	VariantPC or VariantPB	VariantPC or VariantPB
I2cGlobalConfig/I2cChannel/ I2cMasterConfiguration/ I2cHighSpeedModeConfiguration	I2cDataValidDelay	VariantPC or VariantPB	VariantPC or VariantPB
	I2cSetupHoldDelay	VariantPC or VariantPB	VariantPC or VariantPB
	I2cClockHighPeriod	VariantPC or VariantPB	VariantPC or VariantPB

Table continues on the next page...

Integration Manual, Rev. 1.0

Table 8-1.   Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
	I2cClockLowPeriod	VariantPC or VariantPB	VariantPC or VariantPB
	I2cHighSpeedBaudRate	VariantPC or VariantPB	VariantPC or VariantPB
I2cGlobalConfig/I2cChannel/ I2cSlaveConfiguration	I2cSlaveAddress	VariantPC or VariantPB	VariantPC or VariantPB
	I2cSlaveDisableFilterInDoze	VariantPC or VariantPB	VariantPC or VariantPB
	I2cSlaveFilterEnable	VariantPC or VariantPB	VariantPC or VariantPB
	I2cSlaveAckStall	VariantPC or VariantPB	VariantPC or VariantPB
	I2cSlaveTxStall	VariantPC or VariantPB	VariantPC or VariantPB
	I2cSlaveRxStall	VariantPC or VariantPB	VariantPC or VariantPB
	I2cSlaveAdrStall	VariantPC or VariantPB	VariantPC or VariantPB
	I2cGlitchFilterSDA	VariantPC or VariantPB	VariantPC or VariantPB
	I2cGlitchFilterSCL	VariantPC or VariantPB	VariantPC or VariantPB
	I2cDataValidDelay	VariantPC or VariantPB	VariantPC or VariantPB
	I2cClockHoldPeriod	VariantPC or VariantPB	VariantPC or VariantPB
I2cGlobalConfig/I2cChannel/ I2cFlexIOConfiguration	I2cAsyncMethod	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cDmaTxChannelRef	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cDmaRxChannelRef	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cFlexIOSdaPin	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cFlexIOSclPin	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cFlexIOCompareValue	Pre Compile parameter for all Variants of Configuration	Pre Compile
	I2cBaudRate	Pre Compile parameter for all Variants of Configuration	Pre Compile
I2cGlobalConfig/ I2cDemEventParameterRefs	I2c_E_TIMEOUT_FAILURE	Pre Compile parameter for all Variants of Configuration	Pre Compile
CommonPublishedInformation	ArReleaseMajorVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	ArReleaseMinorVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	ArReleaseRevisionVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	ModuleId	Pre Compile parameter for all Variants of Configuration	Pre Compile
	SwMajorVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	SwMinorVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	SwPatchVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile

Table continues on the next page...

Integration Manual, Rev. 1.0

Table 8-1.   Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
	VendorApiInfix	Pre Compile parameter for all Variants of Configuration	Pre Compile
	VendorId	Pre Compile parameter for all Variants of Configuration	Pre Compile

Chapter 9
Integration Steps

This section gives a brief overview of the steps needed for integrating Inter-Integrated Circuit :

- Generate the required I2c configurations. For more details refer to section Files required for Compilation
- Allocate proper memory sections in I2c_MemMap.h and linker command file. For more details refer to section Sections to be defined in [I2c_]MemMap.h
- Compile & build the I2c with all the dependent modules. For more details refer to section Building the Driver

Chapter 10
ISR Reference

ISR functions exported by the I2c driver.

Chapter 11
External Assumptions for I2C driver

The section presents requirements that must be complied with when integrating I2C driver into the application.

N/A