Integration Manual

for S32K1 MCL Driver

Document Number: IM2MCLASR4.4 Rev0000R1.0.1 Rev. 1.0

Chapter 1

Revision History

Revision	Date	Author	Description
1.0	24.02.2022	NXP RTD Team	Prepared for release RTD S32K1 Version 1.0.1

Chapter 2

Introduction

This Integration Manual describes the integration requirements for NXP Semiconductors' Mcl Driver for S32K1.

2.1 Supported Derivatives

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

- s32k116_qfn32
- s32k116_lqfp48
- s32k118_lqfp48
- s32k118_lqfp64
- s32k142_lqfp48
- s32k142_lqfp64
- s32k142_lqfp100
- s32k142w_lqfp48
- s32k142w_lqfp64
- s32k144_lqfp48
- s32k144_lqfp64
- s32k144_lqfp100

- s32k144_mapbga100
- s32k144w_lqfp48
- s32k144w_lqfp64
- s32k146_lqfp64
- s32k146_lqfp100
- s32k146_mapbga100
- s32k146_lqfp144
- s32k148_lqfp100
- s32k148_mapbga100
- s32k148_lqfp144
- s32k148_lqfp176

All of the above microcontroller devices are collectively named as S32K1.

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** style: Used for important terms, notes and warnings.
- *Italic* style: 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.

Warning

This is a warning

2.4 Acronyms and Definitions

Term	Definition
API	Application Programming Interface
ASM	Assembler
BSMI	Basic Software Make file Interface
CAN	Controller Area Network
C/CPP	C and C++ Source Code
CS	Chip Select
CTU	Cross Trigger Unit
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DMA	Direct Memory Access
ECU	Electronic Control Unit
FIFO	First In First Out
LSB	Least Signifigant Bit
MCU	Micro Controller Unit
MIDE	Multi Integrated Development Environment
MSB	Most Significant Bit
N/A	Not Applicable
RAM	Random Access Memory
SIU	Systems Integration Unit
SWS	Software Specification
VLE	Variable Length Encoding
XML	Extensible Markup Language

2.5 Reference List

#	Title	Version
1	S32K1xx Series Reference Manual	Rev. 14, 09/2021
2	Errata S32K116_0N96V	Rev. 22/OCT/2021
3	Errata S32K118_0N97V	Rev. 22/OCT/2021
4	Errata S32K142_0N33V	Rev. 22/OCT/2021
5	Errata S32K144_0N57U	Rev. 22/OCT/2021
6	Errata S32K144W_0P64A	Rev. 22/OCT/2021
7	Errata S32K146_0N73V	Rev. 22/OCT/2021
8	Errata S32K148_0N20V	Rev. 22/OCT/2021
9	S32K1xx Data Sheet	Rev. 14, 08/2021

Chapter 3

Building the driver

- Build Options
- Files required for compilation
- Setting up the plugins

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

3.1 Build Options

- GCC Compiler/Assembler/Linker Options
- GHS Compiler/Assembler/Linker Options
- IAR Compiler/Assembler/Linker Options

The RTD driver files are compiled using:

- NXP GCC 9.2.0 20190812 (Build 1649 Revision gaf57174)
- IAR ANSI C/C++ Compiler V8.40.3.228/W32 for ARM Functional Safety
- Green Hills Multi 7.1.6d / Compiler 2020.1.4

The compiler, assembler, and linker flags used for building the driver are explained below.
The TS_T40D2M10I1R0 part of the plugin name is composed as follows:

- T = Target_Id (e.g. T40 identifies Cortex-M architecture)
- D = Derivative_Id (e.g. D2 identifies S32K1 platform)
- M = SW_Version_Major and SW_Version_Minor
- I = SW_Version_Patch
- R = Reserved

3.1.1 GCC Compiler/Assembler/Linker Options

3.1.1.1 GCC Compiler Options

Compiler Option	Description
-mcpu=cortex-m4	Targeted ARM processor for which GCC should tune the performance of the code (for S32K14x devices)
-mcpu=cortex-m0plus	Targeted ARM processor for which GCC should tune the performance of the code (for S32K11x devices)
-mthumb	Generates code that executes in Thumb state
-mlittle-endian	Generate code for a processor running in little-endian mode
-mfpu=fpv4-sp-d16	Specifies the floating-point hardware available on the target (for S32K14x devices)
-mfloat-abi=hard	Specifies the floating-point ABI to use. "hard" allows generation of floating-point instructions and uses FPU-specific calling conventions (for S32K14x devices)
-mfpu=auto	Specifies the floating-point hardware available on the target (for S32K11x devices)
-mfloat-abi=soft	Specifies the floating-point ABI to use. Specifying "soft" causes GCC to generate output containing library calls for floating-point operations (for S32K11x devices)
-std=c99	Specifies the ISO C99 base standard
-Os	Optimize for size. Enables all -O2 optimizations except those that often increase code size
-ggdb3	Produce debugging information for use by GDB using the most expressive format available, including GDB extensions if at all possible. Level 3 includes extra information, such as all the macro definitions present in the program
-Wall	Enables all the warnings about constructions that some users consider questionable, and that are easy to avoid (or modify to prevent the warning), even in conjunction with macros
-Wextra	This enables some extra warning flags that are not enabled by -Wall
-pedantic	Issue all the warnings demanded by strict ISO C. Reject all programs that use forbidden extensions. Follows the version of the ISO C standard specified by the aforementioend -std option
-Wstrict-prototypes	Warn if a function is declared or defined without specifying the argument types
-Wundef	Warn if an undefined identifier is evaluated in an #if directive. Such identifiers are replaced with zero
-Wunused	Warn whenever a function, variable, label, value, macro is unused
-Werror=implicit-function-declaration	Make the specified warning into an error. This option throws an error when a function is used before being declared
-Wsign-compare	Warn when a comparison between signed and unsigned values could produce an incorrect result when the signed value is converted to unsigned.
-Wdouble-promotion	Give a warning when a value of type float is implicitly promoted to double
-fno-short-enums	Specifies that the size of an enumeration type is at least 32 bits regardless of the size of the enumerator values.

Compiler Option	Description
-funsigned-char	Let the type char be unsigned by default, when the declaration does not use either signed or unsigned
-funsigned-bitfields	Let a bit-field be unsigned by default, when the declaration does not use either signed or unsigned
-fomit-frame-pointer	Omit the frame pointer in functions that dont need one. This avoids the instructions to save, set up and restore the frame pointer; on many targets it also makes an extra register available.
-fno-common	Makes the compiler place uninitialized global variables in the BSS section of the object file. This inhibits the merging of tentative definitions by the linker so you get a multiple-definition error if the same variable is accidentally defined in more than one compilation unit
-fstack-usage	Makes the compiler output stack usage information for the program, on a per-function basis
-fdump-ipa-all	Enables all inter-procedural analysis dumps
-c	Stop after assembly and produce an object file for each source file
-DS32K1XX	Predefine S32K1XX as a macro, with definition 1
-DS32K148	Predefine S32K148 as a macro, with definition 1
-DGCC	Predefine GCC as a macro, with definition 1
-DUSE_SW_VECTOR_MODE	Predefine USE_SW_VECTOR_MODE as a macro, with definition 1. By default, the drivers are compiled to handle interrupts in Software Vector Mode
-DI_CACHE_ENABLE	Predefine I_CACHE_ENABLE as a macro, with definition 1. Enables instruction cache initalization in source file system.c under the Platform driver (for S32K14x devices)
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initalization in source file system.c under the Platform driver (for S32K14x devices)
-DMCAL_ENABLE_USER_MODE_SUPPORT	Predefine MCAL_ENABLE_USER_MODE_SUPPO↩ RT as a macro, with definition 1. Allows drivers to be configured in user mode.

3.1.1.2 GCC Assembler Options

Assembler Option	Description
-Xassembler-with-cpp	Specifies the language for the following input files (rather than letting the compiler choose a default based on the file name suffix)
-mcpu=cortex-m4	Targeted ARM processor for which GCC should tune the performance of the code (for S32K14x devices)
-mcpu=cortex-m0plus	Targeted ARM processor for which GCC should tune the performance of the code (for S32K11x devices)
-mthumb	Generates code that executes in Thumb state
-c	Stop after assembly and produce an object file for each source file

3.1.1.3 GCC Linker Options

Linker Option	Description
-Wl,-Map,filename	Produces a map file
-T linkerfile	Use linkerfile as the linker script. This script replaces the default linker script (rather than adding to it)
–entry=Reset_Handler	Specifies that the program entry point is Reset_Handler
-nostartfiles	Do not use the standard system startup files when linking
-mcpu=cortex-m4	Targeted ARM processor for which GCC should tune the performance of the code (for S32K14x devices)
-mcpu=cortex-m0plus	Targeted ARM processor for which GCC should tune the performance of the code (for S32K11x devices)
-mthumb	Generates code that executes in Thumb state
-mfpu=fpv4-sp-d16	Specifies the floating-point hardware available on the target (for S32K14x devices)
-mfloat-abi=hard	Specifies the floating-point ABI to use. "hard" allows generation of floating-point instructions and uses FPU-specific calling conventions (for S32K14x devices)
-mfpu=auto	Specifies the floating-point hardware available on the target (for S32K11x devices)
-mfloat-abi=soft	Specifies the floating-point ABI to use. Specifying "soft" causes GCC to generate output containing library calls for floating-point operations (for S32K11x devices)
-mlittle-endian	Generate code for a processor running in little-endian mode
-ggdb3	Produce debugging information for use by GDB using the most expressive format available, including GDB extensions if at all possible. Level 3 includes extra information, such as all the macro definitions present in the program
-lc	Link with the C library
-lm	Link with the Math library
-lgcc	Link with the GCC library
-n	Turn off page alignment of sections, and disable linking against shared libraries

3.1.2 GHS Compiler/Assembler/Linker Options

3.1.2.1 GHS Compiler Options

Compiler Option	Description
-cpu=cortexm4	Selects target processor: Arm Cortex M4 (for S32K14x devices)
-cpu=cortexm0plus	Selects target processor: Arm Cortex M0+ (for S32K11x devices)
-thumb	Selects generating code that executes in Thumb state
-fpu=vfpv4_d16	Specifies hardware floating-point using the v4 version of the VFP instruction set, with 16 double-precision floating-point registers (for S32K14x devices)
-fsingle	Use hardware single-precision, software double-precision FP instructions (for S32K14x devices)

Compiler Option	Description
-fsoft	Specifies software floating-point (SFP) mode. This setting causes your target to use integer registers to hold floating-point data and use library subroutine calls to emulate floating-point operations (for S32K11x devices)
-C99	Use (strict ISO) C99 standard (without extensions)
–ghstd=last	Use the most recent version of Green Hills Standard mode (which enables warnings and errors that enforce a stricter coding standard than regular C and C++)
-Osize	Optimize for size
–gnu_asm	Enables GNU extended asm syntax support
-dual_debug	Generate DWARF 2.0 debug information
-G	Generate debug information
-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
-Wimplicit-int	Produce warnings if functions are assumed to return int
-Wshadow	Produce warnings if variables are shadowed
-Wtrigraphs	Produce warnings if trigraphs are detected
-Wundef	Produce a warning if undefined identifiers are used in #if preprocessor statements
–unsigned_chars	Let the type char be unsigned, like unsigned char
–unsigned_fields	Bitfelds declared with an integer type are unsigned
–no_commons	Allocates uninitialized global variables to a section and initializes them to zero at program startup
–no_exceptions	Disables C++ support for exception handling
–no_slash_comment	C++ style // comments are not accepted andgenerate errors
–prototype_errors	Controls the treatment of functions referenced or called when no prototype has been provided
–incorrect_pragma_warnings	Controls the treatment of valid #pragma directives that use the wrong syntax
-c	Stop after assembly and produce an object file for each source file
-DS32K1XX	Predefine S32K1XX as a macro, with definition 1
-DS32K148	Predefine S32K148 as a macro, with definition 1
-DGHS	Predefine GHS as a macro, with definition 1
-DUSE_SW_VECTOR_MODE	Predefine USE_SW_VECTOR_MODE as a macro, with definition 1. By default, the drivers are compiled to handle interrupts in Software Vector Mode
-DI_CACHE_ENABLE	Predefine I_CACHE_ENABLE as a macro, with definition 1. Enables instruction cache initalization in source file system.c under the Platform driver (for S32K14x devices)
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initalization in source file system.c under the Platform driver (for S32K14x devices)

Compiler Option	Description
-DMCAL_ENABLE_USER_MODE_SUPPORT	Predefine MCAL_ENABLE_USER_MODE_SUPPO↩ RT as a macro, with definition 1. Allows drivers to be configured in user mode

3.1.2.2 GHS Assembler Options

Assembler Option	Description
-cpu=cortexm4	Selects target processor: Arm Cortex M4 (for S32K14x devices)
-cpu=cortexm0plus	Selects target processor: Arm Cortex M0+ (for S32K11x devices)
-preprocess_assembly_files	Controls whether assembly files with standard extensions such as .s and .asm are preprocessed
-list	Creates a listing by using the name and directory of the object file with the .lst extension
-c	Stop after assembly and produce an object file for each source file

3.1.2.3 GHS Linker Options

Linker Option	Description
-e Reset_Handler	Make the symbol Reset_Handler be treated as a root symbol and the start label of the application
-T linker_script_file.ld	Use linker_script_file.ld as the linker script. This script replaces the default linker script (rather than adding to it)
-map	Produce a map file
-keepmap	Controls the retention of the map file in the event of a link error
-Mn	Generates a listing of symbols sorted alphabetically/numerically by address
-delete	Instructs the linker to remove functions that are not referenced in the final executable. The linker iterates to find functions that do not have relocations pointing to them and eliminates them
-ignore_debug_references	Ignores relocations from DWARF debug sections when using -delete. DWA↩ RF debug information will contain references to deleted functions that may break some third-party debuggers
-Llibrary_path	Points to library_path (the libraries location) for thumb2 to be used for linking
-larch	Link architecture specific library
-lstartup	Link run-time environment startup routines. The source code for themodules in this library is provided in the src/libstartup directory
-lind_sd	Link language-independent library, containing support routines for features such as software floating point, run-time error checking, C99 complex numbers, and some general purpose routines of the ANSI C library (for S32K14x devices)
-lind_sf	Link language-independent library, containing support routines for features such as software floating point, run-time error checking, C99 complex numbers, and some general purpose routines of the ANSI C library (for S32K11x devices)
-v	Prints verbose information about the activities of the linker, including the libraries it searches to resolve undefined symbols
-keep=C40_Ip_AccessCode	Avoid linker remove function C40_Ip_AccessCode from Fls module because it is not referenced explicitly

Linker Option	Description
-nostartfiles	Controls the start files to be linked into the executable

3.1.3 IAR Compiler/Assembler/Linker Options

3.1.3.1 IAR Compiler Options

Compiler Option	Description
–cpu=Cortex-M4	Targeted ARM processor for which IAR should tune the performance of the code (for S32K14x devices)
–cpu=Cortex-M0+	Targeted ARM processor for which IAR should tune the performance of the code (for S32K11x devices)
–cpu_mode=thumb	Generates code that executes in Thumb state
–endian=little	Generate code for a processor running in little-endian mode
–fpu=FPv4-SP	Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). Single-precision variant. (for S32K14x devices)
–fpu=none	Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). No FPU. (for S32K11x devices)
-e	Enables all IAR C language extensions
-Ohz	Optimize for size. the compiler will emit AEABI attributes indicating the requested optimization goal. This information can be used by the linker to select smaller or faster variants of DLIB library functions
–debug	Makes the compiler include debugging information in the object modules. Including debug information will make the object files larger
–no_clustering	Disables static clustering optimizations. Static and global variables defined within the same module will not be arranged so that variables that are accessed in the same function are close to each other
–no_mem_idioms	Makes the compiler not optimize certain memory access patterns
–no_explicit_zero_opt	Do not treat explicit initializations to zero of static variables as zero initializations
–require_prototypes	Force the compiler to verify that all functions have proper prototypes. Generates an error otherwise
–no_wrap_diagnostics	Does not wrap long lines in diagnostic messages
–diag_suppress=Pa050	Suppresses diagnostic message Pa050
-DS32K1XX	Predefine S32K1XX as a macro, with definition 1
-DS32K148	Predefine S32K148 as a macro, with definition 1
-DIAR	Predefine IAR as a macro, with definition 1
-DUSE_SW_VECTOR_MODE	Predefine USE_SW_VECTOR_MODE as a macro, with definition 1. By default, the drivers are compiled to handle interrupts in Software Vector Mode.

Compiler Option	Description
-DI_CACHE_ENABLE	Predefine I_CACHE_ENABLE as a macro, with definition 1. Enables instruction cache initalization in source file system.c under the Platform driver (for S32K14x devices)
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initalization in source file system.c under the Platform driver (for S32K14x devices)
-DMCAL_ENABLE_USER_MODE_SUPPORT	Predefine MCAL_ENABLE_USER_MODE_SUPPO↩ RT as a macro, with definition 1. Allows drivers to be configured in user mode.

3.1.3.2 IAR Assembler Options

Assembler Option	Description
–cpu=Cortex-M4	Targeted ARM processor for which IAR should tune the performance of the code (for S32K14x devices)
–cpu=Cortex-M0+	Targeted ARM processor for which IAR should tune the performance of the code (for S32K11x devices)
–cpu_mode thumb	Selects the thumb mode for the assembler directive CODE
-g	Disables the automatic search for system include files
-r	Generates debug information

3.1.3.3 IAR Linker Options

Linker Option	Description
–map filename	Produces a map file
–config linkerfile	Use linkerfile as the linker script. This script replaces the default linker script (rather than adding to it)
–cpu=Cortex-M4	Targeted ARM processor for which IAR should tune the performance of the code (for S32K14x devices)
–cpu=Cortex-M0+	Targeted ARM processor for which IAR should tune the performance of the code (for S32K11x devices)
–fpu=FPv4-SP	Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). Single-precision variant. (for S32K14x devices)
–fpu=none	Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). No FPU. (for S32K11x devices)
–entry _start	Treats _start as a root symbol and start label
–enable_stack_usage	Enables stack usage analysis. If a linker map file is produced, a stack usage chapter is included in the map file
–skip_dynamic_initialization	Dynamic initialization (typically initialization of C++ objects with static storage duration) will not be performed automatically during application startup
–no_wrap_diagnostics	Does not wrap long lines in diagnostic messages

3.2 Files required for compilation

This section describes the include files required to compile, assemble and link the Mcl Driver for S32K1 microcontrollers.

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

3.2.0.0.1 MCL Driver Files:

- Mcl_TS_T40D2M10I1R0\src\Cache_Ip.c
- Mcl_TS_T40D2M10I1R0\src\Cache_Ip_HwAcc_Lmem.h
- Mcl_TS_T40D2M10I1R0\src\CDD_Mcl.c
- Mcl_TS_T40D2M10I1R0\src\CDD_Mcl_Ipw.c
- Mcl_TS_T40D2M10I1R0\src\Dma_Ip.c
- Mcl_TS_T40D2M10I1R0\src\Dma_Ip_Driver_State.c
- Mcl_TS_T40D2M10I1R0\src\Dma_Ip_Driver_State.h
- Mcl_TS_T40D2M10I1R0\src\Dma_Ip_Hw_Access.c
- Mcl_TS_T40D2M10I1R0\src\Dma_Ip_Hw_Access.h
- Mcl_TS_T40D2M10I1R0\src\Dma_Ip_Hwv2_AccessInline.h
- Mcl_TS_T40D2M10I1R0\src\Dma_Ip_Irq.c
- Mcl_TS_T40D2M10I1R0\src\Dma_Ip_Multicore.c
- Mcl_TS_T40D2M10I1R0\src\Flexio_Mcl_Ip.c
- Mcl_TS_T40D2M10I1R0\src\Flexio_Mcl_Ip_HwAccess.c
- Mcl_TS_T40D2M10I1R0\src\Flexio_Mcl_Ip_Irq.c
- Mcl_TS_T40D2M10I1R0\src\Trgmux_Ip.c
- Mcl_TS_T40D2M10I1R0\src\Trgmux_Ip_HwAcc.h

- Mcl_TS_T40D2M10I1R0\include\Cache_Ip.h
- Mcl_TS_T40D2M10I1R0\include\Cache_Ip_Devassert.h
- Mcl_TS_T40D2M10I1R0\include\Cache_Ip_Types.h
- Mcl_TS_T40D2M10I1R0\include\CDD_Mcl.h
- Mcl_TS_T40D2M10I1R0\include\CDD_Mcl_Ipw.h
- Mcl_TS_T40D2M10I1R0\include\CDD_Mcl_Irq.h
- Mcl_TS_T40D2M10I1R0\include\Dma_Ip.h

- Mcl_TS_T40D2M10I1R0\include\Dma_Ip_Devassert.h
- Mcl_TS_T40D2M10I1R0\include\Dma_Ip_Hwv3_Regs.h
- Mcl_TS_T40D2M10I1R0\include\Dma_Ip_Irq.h
- Mcl_TS_T40D2M10I1R0\include\Dma_Ip_Multicore.h
- Mcl_TS_T40D2M10I1R0\include\Dma_Ip_Types.h
- Mcl_TS_T40D2M10I1R0\include\Flexio_Mcl_Ip.h
- Mcl_TS_T40D2M10I1R0\include\Flexio_Mcl_Ip_HwAccess.h
- Mcl_TS_T40D2M10I1R0\include\Flexio_Mcl_Ip_Types.h
- Mcl_TS_T40D2M10I1R0\include\Mcl.h
- Mcl_TS_T40D2M10I1R0\include\Mcl_Types.h
- Mcl_TS_T40D2M10I1R0\include\Trgmux_Ip.h
- Mcl_TS_T40D2M10I1R0\include\Trgmux_Ip_Devassert.h
- Mcl_TS_T40D2M10I1R0\include\Trgmux_Ip_Types.h

3.2.0.0.2 MCL Driver Generated Files (must be generated by the user using a configuration tool):

- Cache_Ip_Cfg_Defines.h
- Cache_Ip_Cfg_DeviceRegisters.h
- CDD_Mcl_Cfg.h
- CDD_Mcl_Cfg_Defines.h
- Dma_Ip_Cfg.h
- Dma_Ip_Cfg_Defines.h
- Dma_Ip_Cfg_DeviceRegistersV2.h
- Dma_Ip_Cfg_Devices.h
- Flexio_Mcl_Ip_Cfg.h
- Flexio_Mcl_Ip_Cfg_Defines.h
- Flexio_Mcl_Ip_Cfg_DeviceRegisters.h
- Trgmux_Ip_Cfg.h
- Trgmux_Ip_Cfg_Defines.h
- Trgmux_Ip_Cfg_DeviceRegisters.h
- CDD_Mcl_Cfg.c
- Dma_Ip_Cfg.c
- Trgmux_Ip_Cfg.c

Note

As a deviation from the standard:

- Eth_[VariantName]_PBcfg.c, Eth_Ipw_[VariantName]_PBcfg.c, Gmac_Ip_[VariantName]_PBcfg.c - These files will contain the definition for all parameters that are variant aware, independent of the configuration class that will be selected (PC, LT, PB)
- Gmac_Ip_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 Eth_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 VariantPreCompile.

3.2.0.0.3 Base Files:

- Base_TS_T40D2M10I1R0\include\Mcal.h
- Base_TS_T40D2M10I1R0\include\Platform_Types.h
- Base_TS_T40D2M10I1R0\include\Soc_Ips.h
- Base_TS_T40D2M10I1R0\include\Std_Types.h
- Base_TS_T40D2M10I1R0\include\OsIf.h
- Base_TS_T40D2M10I1R0\generate_PC\include\modules.h

3.2.0.0.4 DEM Files:

- 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

3.2.0.0.5 DET Files:

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

3.2.0.0.6 RTE Files:

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

3.3 Setting up the plugins

The *driver* driver was designed to be configured by using the EB Tresos Studio (version EB tresos Studio *version* or later.)

Chapter 4

Function calls to module

4.1 Function Calls during Start-up

None.

4.2 Function Calls during Shutdown

None.

4.3 Function Calls during Wake-up

None.

Chapter 5

Module requirements

- Exclusive areas to be defined in BSW scheduler
- unavailable_exclusive_areas
- Peripheral Hardware Requirements
- ISR to configure within AutosarOS - dependencies
- ISR Macro
- Other AUTOSAR modules - dependencies
- Data Cache Restrictions
- User Mode support
- Multicore support

5.1 Exclusive areas to be defined in BSW scheduler

In the current implementation, MCL is using the services of Schedule Manager (SchM) for entering and exiting the exclusive areas. The following critical regions are used in the MCL driver:

Exclusive Areas are used in High level driver layer (HLD)

MCL_EXCLUSIVE_AREA_00 is used in function Mcl_Init to protect the DMA_MP_CSR and DMA_C↩ RC_GEC register from read/modify/write operation in Dma_Ip_Init.

MCL_EXCLUSIVE_AREA_01 is used in function Mcl_DeInit to protect the DMA_MP_CSR and DMA↩ _CRC_GEC register from read/modify/write operation in Dma_Ip_Deinit.

MCL_EXCLUSIVE_AREA_02 is used in function Mcl_SetDmaInstanceCommand to protect DMA_MP_↩ CSR register from read/modify/write operation in Dma_Ip_SetLogicInstanceCommand.

MCL_EXCLUSIVE_AREA_03 is used in function Mcl_Init to protect DMA_CRC_CTL register from read/modify/write operation in Dma_Ip_LogicChannelInit.

MCL_EXCLUSIVE_AREA_04 is used in function Mcl_DeInit to protect All TCDx_WORDs, DMA_MP_↩ CH_GRPRI and DMAMUX_CHCFG register from read/modify/write operation in Dma_Ip_LogicChannelDeinit.

MCL_EXCLUSIVE_AREA_05 is used in function Mcl_SetDmaChannelCommand to protect TCDx.8TH_↩ WORD register from read/modify/write operation in Dma_Ip_SetLogicChannelCommand.

MCL_EXCLUSIVE_AREA_06 is used in function Mcl_GetDmaChannelStatus to protect TCDx.8TH_W↩ ORD register from read/modify/write operation in Dma_Ip_GetLogicChannelStatus.

MCL_EXCLUSIVE_AREA_07 is used in function Mcl_SetDmaChannelGlobalList to protect DMA_MP_↩ CH_GRPRI, DMAMUX_CHCFG and TCDx.8TH_WORD register from read/modify/write operation in Dma_↩ Ip_SetLogicChannelGlobalList.

MCL_EXCLUSIVE_AREA_08 is used in function Mcl_SetDmaChannelGlobalList to protect All TCDx_↩ WORDs register from read/modify/write operation in Dma_Ip_SetLogicChannelGlobalList.

MCL_EXCLUSIVE_AREA_09 is used in function Mcl_SetDmaChannelScatterGatherList to protect All T↩ CDx_WORDs register from read/modify/write operation in Dma_Ip_SetLogicChannelScatterGatherList.

MCL_EXCLUSIVE_AREA_10 is used in function Mcl_SetDmaChannelScatterGatherList to protect All T↩ CDx_WORDs register from read/modify/write operation in Dma_Ip_SetLogicChannelScatterGatherList.

MCL_EXCLUSIVE_AREA_11 is used in function Mcl_SetDmaChannelScatterGatherConfig to protect All TCDx_WORDs register from read/modify/write operation in Dma_Ip_SetLogicChannelScatterGatherConfig.

MCL_EXCLUSIVE_AREA_12 is used in function Mcl_Init to protect All TCDx_WORDs register from read/modify/write operation in Static_Dma_Ip_SetLogicChannelScatterGatherInit.

MCL_EXCLUSIVE_AREA_13 is used in function Mcl_CacheEnable to protect S32_SCB_CCR register from read/modify/write operation in Cache_Ip_Enable.

MCL_EXCLUSIVE_AREA_14 is used in function Mcl_CacheDisable to protect S32_SCB_CCR register from read/modify/write operation in Cache_Ip_Disable.

MCL_EXCLUSIVE_AREA_15 is used in function Mcl_CacheInvalidate to protect S32_SCB_CSSELR and S32_SCB_ICIALLU register from read/modify/write operation in Cache_Ip_Invalidate.

MCL_EXCLUSIVE_AREA_16 is used in function Mcl_CacheClean to protect S32_SCB_CSSELR and S32↩ _SCB_ICIALLU register from read/modify/write operation in Cache_Ip_Clean.

MCL_EXCLUSIVE_AREA_17 is used in function Mcl_CacheInvalidateByAddr to protect S32_SCB_CSS↩ ELR and S32_SCB_ICIMVAU register from read/modify/write operation in Cache_Ip_InvalidateByAddr.

MCL_EXCLUSIVE_AREA_18 is used in function Mcl_CacheCleanByAddr to protect S32_SCB_CSSELR and S32_SCB_ICIMVAU register from read/modify/write operation in Cache_Ip_CleanByAddr.

MCL_EXCLUSIVE_AREA_19 is used in function Mcl_Init to protect TRGMUXn register from read/modify/write operation in Trgmux_Ip_Init.

MCL_EXCLUSIVE_AREA_20 is used in function Mcl_SetTrgMuxInput to protect TRGMUXn register from read/modify/write operation in Trgmux_Ip_SetInput.

MCL_EXCLUSIVE_AREA_21 is used in function Mcl_SetTrgMuxLock to protect TRGMUXn register from read/modify/write operation in Trgmux_Ip_SetLock.

MCL_EXCLUSIVE_AREA_39 is used in function Mcl_Init to protect CTRL register from read/modify/write operation in Flexio_Mcl_Ip_InitDevice.

MCL_EXCLUSIVE_AREA_40 is used in function Mcl_Init to protect CTRL register from read/modify/write operation in Flexio_Mcl_Ip_InitDevice.

MCL_EXCLUSIVE_AREA_41 is used in function Mcl_Init to protect CTRL register from read/modify/write operation in Flexio_Mcl_Ip_InitDevice.

MCL_EXCLUSIVE_AREA_42 is used in function Mcl_Init to protect SHIFTEIEN register from read/modify/write operation in Flexio_Mcl_Ip_InitDevice.

MCL_EXCLUSIVE_AREA_43 is used in function Mcl_Init to protect SHIFTSIEN register from read/modify/write operation in Flexio_Mcl_Ip_InitDevice.

MCL_EXCLUSIVE_AREA_44 is used in function Mcl_Init to protect SHIFTSDEN register from read/modify/write operation in Flexio_Mcl_Ip_InitDevice.

MCL_EXCLUSIVE_AREA_45 is used in function Mcl_Init to protect TIMIEN register from read/modify/write operation in Flexio_Mcl_Ip_InitDevice.

MCL_EXCLUSIVE_AREA_46 is used in function Mcl_Init to protect TIMERSDEN register from read/modify/write operation in Flexio_Mcl_Ip_InitDevice.

Exclusive Areas implemented in Low level driver layer (IPL)

MCL_EXCLUSIVE_AREA_00 is used in function Dma_Ip_Init() to protect the updates for:

- DMA_MP_CSR
- DMA_CRC_GEC

MCL_EXCLUSIVE_AREA_01 is used in function Dma_Ip_Deinit() to protect the updates for:

- DMA_MP_CSR
- DMA_CRC_GEC

MCL_EXCLUSIVE_AREA_02 is used in function Dma_Ip_SetLogicInstanceCommand() to protect the updates for:

- DMA_MP_CSR

MCL_EXCLUSIVE_AREA_03 is used in function Dma_Ip_LogicChannelInit() to protect the updates for:

- DMA_CRC_CTL

MCL_EXCLUSIVE_AREA_04 is used in function Dma_Ip_LogicChannelDeinit() to protect the updates for:

- All TCDx_WORDs
- DMA_MP_CH_GRPRI
- DMAMUX_CHCFG

MCL_EXCLUSIVE_AREA_05 is used in function Dma_Ip_SetLogicChannelCommand() to protect the updates for:

- TCDx.8TH_WORD

MCL_EXCLUSIVE_AREA_06 is used in function Dma_Ip_GetLogicChannelStatus() to protect the updates for:

- TCDx.8TH_WORD

MCL_EXCLUSIVE_AREA_07 is used in function Dma_Ip_SetLogicChannelGlobalList() to protect the updates for:

- DMA_MP_CH_GRPRI
- DMAMUX_CHCFG
- TCDx.8TH_WORD

MCL_EXCLUSIVE_AREA_08 is used in function Dma_Ip_SetLogicChannelTransferList() to protect the updates for:

- All TCDx_WORDs

MCL_EXCLUSIVE_AREA_09 is used in function Dma_Ip_SetLogicChannelScatterGatherList() to protect the updates for:

- All TCDx_WORDs

MCL_EXCLUSIVE_AREA_10 is used in function Dma_Ip_SetLogicChannelCrcList() to protect the updates for:

- DMA_CRC_CTL
- DMA_CRC_ICRC

MCL_EXCLUSIVE_AREA_11 is used in function Dma_Ip_SetLogicChannelScatterGatherConfig() to protect the updates for:

- All TCDx_WORDs

MCL_EXCLUSIVE_AREA_12 is used in function Static_Dma_Ip_SetLogicChannelScatterGatherInit() to protect the updates for:

- All TCDx_WORDs

MCL_EXCLUSIVE_AREA_13 is used in function Cache_Ip_Enable() to protect the updates for:

- S32_SCB_CCR

MCL_EXCLUSIVE_AREA_14 is used in function Cache_Ip_Disable() to protect the updates for:

- S32_SCB_CCR

MCL_EXCLUSIVE_AREA_15 is used in function Cache_Ip_Invalidate() to protect the updates for:

- S32_SCB_CSSELR
- S32_SCB_ICIALLU

MCL_EXCLUSIVE_AREA_16 is used in function Cache_Ip_Clean() to protect the updates for:

- S32_SCB_CSSELR
- S32_SCB_ICIALLU

MCL_EXCLUSIVE_AREA_17 is used in function Cache_Ip_InvalidateByAddr() to protect the updates for:

- S32_SCB_CSSELR
- S32_SCB_ICIMVAU

MCL_EXCLUSIVE_AREA_18 is used in function Cache_Ip_CleanByAddr() to protect the updates for:

- S32_SCB_CSSELR
- S32_SCB_ICIMVAU

MCL_EXCLUSIVE_AREA_19 is used in function Trgmux_Ip_Init() to protect the updates for:

- TRGMUXn

MCL_EXCLUSIVE_AREA_20 is used in function Trgmux_Ip_SetInput() to protect the updates for:

- TRGMUXn

MCL_EXCLUSIVE_AREA_21 is used in function Trgmux_Ip_SetLock() to protect the updates for:

- TRGMUXn

MCL_EXCLUSIVE_AREA_39 is used in function Flexio_Mcl_Ip_SetSoftwareReset() to protect the updates for:

- CTRL

MCL_EXCLUSIVE_AREA_40 is used in function Flexio_Mcl_Ip_SetDebugEnable() to protect the updates for:

- CTRL

MCL_EXCLUSIVE_AREA_41 is used in function Flexio_Mcl_Ip_SetEnable() to protect the updates for:

- CTRL

MCL_EXCLUSIVE_AREA_42 is used in function Flexio_Mcl_Ip_SetShifterErrorInterrupt() to protect the updates for:

- SHIFTEIEN

MCL_EXCLUSIVE_AREA_43 is used in function Flexio_Mcl_Ip_SetShifterInterrupt() to protect the updates for:

- SHIFTSIEN

MCL_EXCLUSIVE_AREA_44 is used in function Flexio_Mcl_Ip_SetShifterDMARequest() to protect the updates for:

- SHIFTSDEN

MCL_EXCLUSIVE_AREA_45 is used in function Flexio_Mcl_Ip_SetTimerInterrupt() to protect the updates for:

- TIMIEN

MCL_EXCLUSIVE_AREA_46 is used in function Flexio_Mcl_Ip_SetTimerDMARequest() to protect the updates for:

- IMERSDEN

Table 5.1 Exclusive Areas

#	AREA_00	AREA_01	AREA_02	AREA_03	AREA_04	AREA_05	AREA_06	AREA_07	AREA_08	AREA_09	AREA_10	AREA_11	AREA_12	AREA_13	AREA_14	AREA_15	AREA_16	AREA_17	AREA_18	AREA_19	AREA_20	AREA_21	AREA_39	AREA_40	AREA_41	AREA_42	AREA_43	AREA_44	AREA_45	AREA_46
AREA_00	x	x	x																											
AREA_01	x	x	x																											
AREA_02	x	x	x																											
AREA_03				x							x																			
AREA_04					x	x		x	x	x		x	x																	
AREA_05					x	x		x	x	x		x	x																	
AREA_06					x	x		x	x	x		x	x																	

#	ARE A_00	ARE A_01	ARE A_02	ARE A_03	ARE A_04	ARE A_05	ARE A_06	ARE A_07	ARE A_08	ARE A_09	ARE A_10	ARE A_11	ARE A_12	ARE A_13	ARE A_14	ARE A_15	ARE A_16	ARE A_17	ARE A_18	ARE A_19	ARE A_20	ARE A_21	ARE A_39	ARE A_40	ARE A_41	ARE A_42	ARE A_43	ARE A_44	ARE A_45	ARE A_46
ARE A_07					x	x		x	x	x		x	x																	
ARE A_08				x							x																			
ARE A_09					x	x		x	x	x		x	x																	
ARE A_10					x	x		x	x	x		x	x																	
ARE A_11																														
ARE A_12																														
ARE A_13																														

#	ARE↩A↩_↩00	ARE↩A↩_↩01	ARE↩A↩_↩02	ARE↩A↩_↩03	ARE↩A↩_↩04	ARE↩A↩_↩05	ARE↩A↩_↩06	ARE↩A↩_↩07	ARE↩A↩_↩08	ARE↩A↩_↩09	ARE↩A↩_↩10	ARE↩A↩_↩11	ARE↩A↩_↩12	ARE↩A↩_↩13	ARE↩A↩_↩14	ARE↩A↩_↩15	ARE↩A↩_↩16	ARE↩A↩_↩17	ARE↩A↩_↩18	ARE↩A↩_↩19	ARE↩A↩_↩20	ARE↩A↩_↩21	ARE↩A↩_↩39	ARE↩A↩_↩40	ARE↩A↩_↩41	ARE↩A↩_↩42	ARE↩A↩_↩43	ARE↩A↩_↩44	ARE↩A↩_↩45	ARE↩A↩_↩46
ARE↩A↩_↩14																														
ARE↩A↩_↩15																														
ARE↩A↩_↩16																														
ARE↩A↩_↩17																														
ARE↩A↩_↩18																														
ARE↩A↩_↩19																				x	x	x								
ARE↩A↩_↩20																				x	x	x								

#	ARE_A_00	ARE_A_01	ARE_A_02	ARE_A_03	ARE_A_04	ARE_A_05	ARE_A_06	ARE_A_07	ARE_A_08	ARE_A_09	ARE_A_10	ARE_A_11	ARE_A_12	ARE_A_13	ARE_A_14	ARE_A_15	ARE_A_16	ARE_A_17	ARE_A_18	ARE_A_19	ARE_A_20	ARE_A_21	ARE_A_39	ARE_A_40	ARE_A_41	ARE_A_42	ARE_A_43	ARE_A_44	ARE_A_45	ARE_A_46
ARE_A_21																				x	x	x								
ARE_A_39																							x	x	x					
ARE_A_40																							x	x	x					
ARE_A_41																							x	x	x					
ARE_A_42																										x				
ARE_A_43																											x			
ARE_A_44																												x		

#	ARE_A_00	ARE_A_01	ARE_A_02	ARE_A_03	ARE_A_04	ARE_A_05	ARE_A_06	ARE_A_07	ARE_A_08	ARE_A_09	ARE_A_10	ARE_A_11	ARE_A_12	ARE_A_13	ARE_A_14	ARE_A_15	ARE_A_16	ARE_A_17	ARE_A_18	ARE_A_19	ARE_A_20	ARE_A_21	ARE_A_39	ARE_A_40	ARE_A_41	ARE_A_42	ARE_A_43	ARE_A_44	ARE_A_45	ARE_A_46
ARE_A_45																													x	
ARE_A_46																														x

5.2 Peripheral Hardware Requirements

None.

5.3 ISR to configure within AutosarOS - dependencies

The following ISRs are used by the Mcl Driver when interrupts are switched on (the driver can also be run in polling mode):

ISR Name	NVIC Interrupt ID
DMA0_IRQn	0
DMA1_IRQn	1
DMA2_IRQn	2
DMA3_IRQn	3
DMA4_IRQn	4
DMA5_IRQn	5
DMA6_IRQn	6
DMA7_IRQn	7
DMA8_IRQn	8
DMA9_IRQn	9
DMA10_IRQn	10
DMA11_IRQn	11

ISR Name	NVIC Interrupt ID
DMA12_IRQn	12
DMA13_IRQn	13
DMA14_IRQn	14
DMA15_IRQn	15
DMA_Error_IRQn	16

5.4 ISR Macro

RTD 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.

5.4.1 Without an Operating System

The macro *USING_OS_AUTOSAROS* must not be defined.

5.4.1.1 Using Software Vector Mode

The macro *USE_SW_VECTOR_MODE* must be defined and the ISR macro is defined as:

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

In this case, the drivers' interrupt handlers are normal C functions and their prologue/epilogue will handle the context save and restore.

5.4.1.2 Using Hardware Vector Mode

The macro *USE_SW_VECTOR_MODE* must not defined and the ISR macro is defined as:

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

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

5.4.2 With an Operating System

Please refer to your OS documentation for description of the ISR macro.

5.5 Other AUTOSAR modules - dependencies

- Mcu: The Microcontroller Unit Driver (MCU Driver) is primarily responsible for initializing and controlling the chips internal clock sources and clock prescalers. The clock frequency may affect the Trigger frequency, Conversion time and Sampling time.
- Det: If development error detection for the MCL module is enabled: The MCL module shall raise errors to the Development Error Tracer (DET) whenever a development error is encountered by this module.
- Base: The Base module contains the common files/definitions needed by all RTD modules.
- Resource: is required to select processor derivative. Current MCL driver has support for the following derivatives, everyone having attached a Resource file: s32g233a_bga525, s32g234m_bga525, s32g254a_bga525, s32g274a_bga525, s32r45_bga780.
- Rte: Used to manage the exclusive area inside MCL module.
- EcuC: This module is required for configuring the variant handling in Tresos.
- Os: This module is required for configuring the Partition mapping with core ID in Tresos.
- Plarform: This module is required for configuring the Interrupt controller in Tresos.

5.6 Data Cache Restrictions

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 _driver___Memmap).

5.7 User Mode support

5.7.1 User Mode configuration in the module

The Mcl can be run in user mode if the following steps are performed:

- Enable **MclEnableUserModeSupport** from the configuration
- Call the following functions as trusted functions:

Function syntax	Description	Available via
void Mcl_Dma_SetUserAccess↩ Allowed(void)	For seting the user access allowed for DMA registers protected by REG↩ _PROT	Dma_Ip_TrustedFunctions.h
LOCAL_INLINE Std_ReturnType hwAcc_Lmem_ProcessorCode↩ CacheEnable(void)	Enable Instruction Cache	

Function syntax	Description	Available via
LOCAL_INLINE Std_ReturnType hwAcc_Lmem_ProcessorCode↩ CacheDisable(void)	Disable Instruction Cache	
LOCAL_INLINE Std_ReturnType hwAcc_Lmem_ProcessorCode↩ Invalidate(void)	Invalidate Instruction Cache	
LOCAL_INLINE Std_ReturnType hwAcc_Lmem_ProcessorCode↩ Clean(const boolean enInvalidate)	Clean Instruction Cache	
LOCAL_INLINE Std_ReturnType hwAcc_Lmem_ProcessorCode↩ InvalidateByAddr(const uint32 addr, const uint32 length)	Invalidate Instruction Cache By Address	
LOCAL_INLINE Std_ReturnType hwAcc_Lmem_ProcessorCode↩ CacheCleanByAddr(const boolean enInvalidate, const uint32 addr, const uint32 length)	Clean Instruction Cache By Address	
Trgmux_Ip_StatusType hwAcc_↩ Init(TRGMUX_Type * const p↩ Trgmux)	Initialize the Trgmux	Trgmux_Ip_TrustedFunctions.h
void hwAcc_SetInputForOutput(↩ TRGMUX_Type * const pTrgmux, const uint32 Input, const uint32 Output)	Mapping the Input for the Output	
void hwAcc_SetLockForOutput(↩ TRGMUX_Type * const pTrgmux, const uint32 Output)	Lock the Output	
boolean hwAcc_GetLockFor↩ Output(const TRGMUX_Type * const pTrgmux, const uint32 Output)	Get Lock status for the Output	
void Ftm_Mcl_Ip_SetUser↩ AccessAllowed(void)	For seting the user access allowed for FTM registers protected by REG↩ _PROT	Ftm_Ip_TrustedFunctions.h

Note: All of Cache functions are static inline function to avoid using stack. Because cache will work incorrectly when using stack. If user want to use cache with stack, the stack have to be pushed into non-cache memory section.

5.7.2 User Mode configuration in AutosarOS

When User mode is enabled, the driver may has the functions that need to be called as trusted functions in AutosarOS context. Those functions are already defined in driver and declared in the header `<IpName>_Ip↩ _TrustedFunctions.h`. This header also included all headers files that contains all types definition used by parameters or return types of those functions. Refer the chapter User Mode configuration in the module for more detail about those functions and the name of header files they are declared inside. Those functions will be called indirectly with the naming convention below in order to AutosarOS can call them as trusted functions.

```
Call_<Function_Name>_TRUSTED(parameter1,parameter2,...)
```

That is the result of macro expansion `OsIf_Trusted_Call` in driver code:

`#define` OsIf_Trusted_Call[1-6params](name,param1,...,param6) Call_##name##_TRUSTED(param1,...,param6)

So, the following steps need to be done in AutosarOS:

- Ensure `MCAL_ENABLE_USER_MODE_SUPPORT` macro is defined in the build system or somewhere global.
- Define and declare all functions that need to call as trusted functions follow the naming convention above in Integration/User code. They need to visible in `Os.h` for the driver to call them. They will do the marshalling of the parameters and call `CallTrustedFunction()` in OS specific manner.
- `CallTrustedFunction()` will switch to privileged mode and call `TRUSTED_<Function_Name>()`.
- `TRUSTED_<Function_Name>()` function is also defined and declared in Integration/User code. It will unmarshalling of the parameters to call `<Function_Name>`() of driver. The `<Function_Name>`() functions are already defined in driver and declared in `<IpName>_Ip_TrustedFunctions.h`. This header should be included in OS for OS call and indexing these functions.

See the sequence chart below for an example calling `Linflexd_Uart_Ip_Init_Privileged()` as a trusted function.

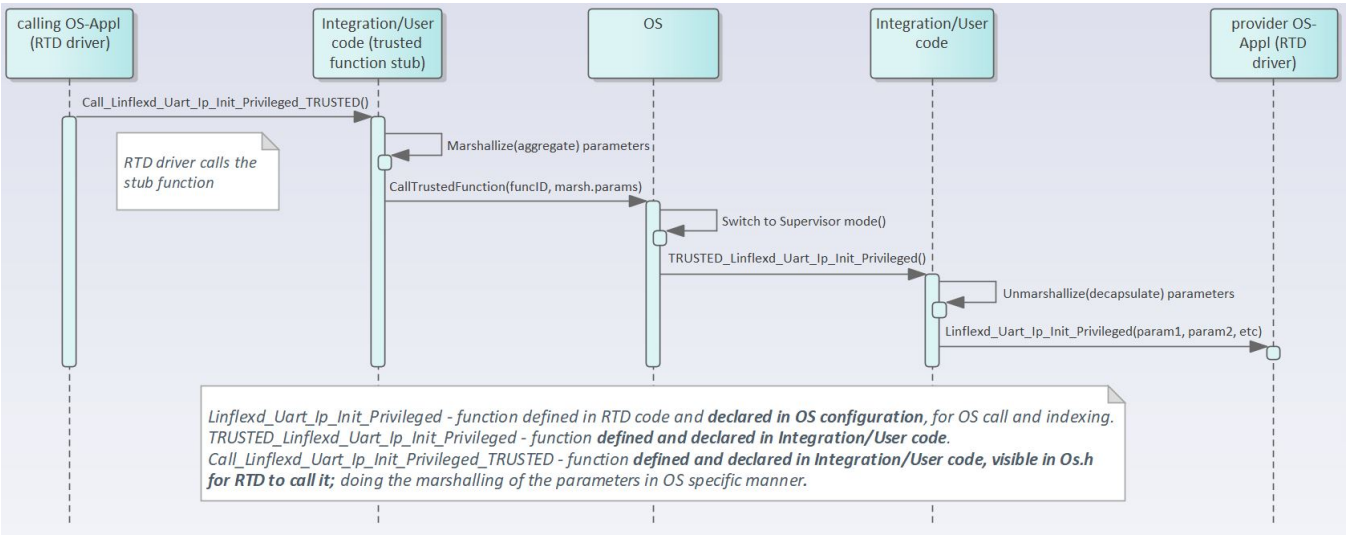

Figure 5.1 Example sequence chart for calling `Linflexd_Uart_Ip_Init_Privileged` as trusted function

5.8 Multicore support

The Mcl Driver don't have Multicore Support for S32K1XX.

Chapter 6

Main API Requirements

- Main function calls within BSW scheduler
- API Requirements
- Calls to Notification Functions, Callbacks, Callouts

6.1 Main function calls within BSW scheduler

None.

6.2 API Requirements

API: “void Mcl_SetDmaInstanceCommand(const uint32 Instance, const Mcl_DmaInstanceCmdType Command)”

The Dma Instance Command is used to trigger a specific action by the DMA Engine. The command is specific and it shall be selected from the “Mcl_DmaInstanceCmdType” enumeration. Example: If the user desires to pause a specific Logic Instance during runtime, the “Mcl_DmaInst_Pause” command shall be used.

API: “void Mcl_GetDmaInstanceStatus(const uint32 Instance, Mcl_DmaInstanceStatusType * const Status)”

The Dma Instance Status is used to get the specific status of the DMA Engine. The status contains hardware information for the selected Logic Instance.

API: “void Mcl_SetDmaChannelCommand(const uint32 Channel, const Mcl_DmaChannelCmdType Command)”

The Dma Channel Command is used to trigger a specific action by the DMA Engine regarding the configured TCD. The command is specific and it shall be selected from the “Mcl_DmaChannelCmdType” enumeration. Example: If the user desires to start the reception of a specific hardware signal, that triggers the Logic Channel, during runtime, the “Mcl_DmaCh_StartRequest” command shall be used.

API: “void Mcl_GetDmaChannelStatus(const uint32 Channel, Mcl_DmaChannelStatusType * const Status)”

The Dma Instance Status is used to get the specific status of the DMA TCD. The status contains hardware information for the selected Logic Channel.

API: “Mcl_SetDmaChannelGlobalList(const uint32 Channel, const Mcl_DmaChannelGlobalListType List[], const uint32 ListDimension)”

The Dma Channel can be configured during runtime, in case that the generated configuration by Tresos does not cover all the use cases. The Global List configuration configures Logic Channel values, like: Global Control Settings, Request Settings, Interrupt Settings and Priority Settings. The user shall create Global Configuration Lists (“↩ Mcl_DmaChannelGlobalListType”) based on the required configuration. EXAMPLE: The user needs to configure the Logic Channel Global Configuration Parameters: “Mcl_DmaSetRequest_MuxSource”, “Mcl_DmaSetRequest↩ _EnMuxSource”, “Mcl_DmaSetPriority_Level” and “Mcl_DmaSetPriority_EnPreemption”.

The order shall be:

Step0: Disable the Mux Source.

Step1: Change the Mux Source Value.

Step2: Change the Priority Level Value.

Step3: Enable the Preemption.

Step4: Enable the Mux Source.

CODE:

```
#define DMA_DSPI0_TX_GLOBAL_CFG0 ((uint32)5U)

Mcl_DmaChannelGlobalListType DmaChannelSpiTxGlobalCfg0[DMA_DSPI0_TX_GLOBAL_CFG0];

DmaChannelSpiTxGlobalCfg0[0U].Param = Mcl_DmaSetRequest_EnMuxSource;

DmaChannelSpiTxGlobalCfg0[0U].Value = FALSE;

DmaChannelSpiTxGlobalCfg0[1U].Param = Mcl_DmaSetRequest_MuxSource;

DmaChannelSpiTxGlobalCfg0[1U].Value = DMA_IP_REQ_MUX0_DSPI0_TX;

DmaChannelSpiTxGlobalCfg0[2U].Param = Mcl_DmaSetPriority_Level;

DmaChannelSpiTxGlobalCfg0[2U].Value = DMA_IP_LEVEL_PRIO0;

DmaChannelSpiTxGlobalCfg0[3U].Param = Mcl_DmaSetPriority_EnPreemption;

DmaChannelSpiTxGlobalCfg0[3U].Value = TRUE;

DmaChannelSpiTxGlobalCfg0[4U].Param = Mcl_DmaSetRequest_EnMuxSource;

DmaChannelSpiTxGlobalCfg0[4U].Value = TRUE;

Mcl_SetDmaChannelGlobalList(CHANNEL_DSPI0_TX, &DmaChannelSpiTxGlobalCfg0[0U], DMA_DSPI0_TX_GLOBAL_
```

API: “void Mcl_SetDmaChannelTransferList(const uint32 Channel, const Mcl_DmaChannelTransferListType List[], const uint32 ListDimension)”

The Dma Channel can be configured during runtime, in case that the generated configuration by Tresos does not cover all the use cases. The Transfer List configuration configures Logic Channel Hardware TCD values, like: Transfer Control Settings, Source Settings, Destination Settings, MinorLoop Settings and MajorLoop Settings. The user shall create Transfer Configuration Lists (“Mcl_DmaChannelTransferListType”) based on the required configuration. The Transfer Configuration List shall automatically change the Logic Channel configuration from Scatter/Gather Mode to Transfer Mode. EXAMPLE: The user needs to configure the Logic Channel Transfer Configuration Parameters: “Mcl_DmaSetSource_Address”, “Mcl_DmaSetSource_SignedOffset”, “Mcl_DmaSetSource_TransferSize”, “Mcl_DmaSetDestination_Address”, “Mcl_DmaSetDestination_SignedOffset”, “Mcl_DmaSetDestination_TransferSize”, “Mcl_DmaSetMinorLoop_Size”, “Mcl_DmaSetMajorLoop_Count”, “Mcl_DmaSetControl_BandwidthControl”.

The order shall be:

Step0: Change the Source Address Value.

Step1: Change the Source Signed Offset Value.

Step2: Change the Source Transfer Size Value.

Step3: Change the Destination Address Value.

Step4: Change the Destination Signed Offset Value.

Step5: Change the Destination Transfer Size Value.

Step6: Change the Minor Loop Size Value.

Step7: Change the Major Loop Count Value.

Step8: Change the Bandwidth Value.

CODE:

```
#define DMA_DSPI0_TX_TRANSFER_CFG0 ((uint32)9U)

Mcl_DmaChannelTransferListType DmaChannelSpiTxTransferCfg0[DMA_DSPI0_TX_TRANSFER_CFG0];

DmaChannelSpiTxTransferCfg0[0U].Param = Mcl_DmaSetSource_Address;

DmaChannelSpiTxTransferCfg0[0U].Value = &SpiTx_Buffer0;

DmaChannelSpiTxTransferCfg0[1U].Param = Mcl_DmaSetSource_SignedOffset;

DmaChannelSpiTxTransferCfg0[1U].Value = 4U;

DmaChannelSpiTxTransferCfg0[2U].Param = Mcl_DmaSetSource_TransferSize;

DmaChannelSpiTxTransferCfg0[2U].Value = DMA_IP_TRANSFER_SIZE_2_BYTE;

DmaChannelSpiTxTransferCfg0[3U].Param = Mcl_DmaSetDestination_Address;

DmaChannelSpiTxTransferCfg0[3U].Value = &SPI_TX_REG;

DmaChannelSpiTxTransferCfg0[4U].Param = Mcl_DmaSetDestination_SignedOffset;
```

DmaChannelSpiTxTransferCfg0[4U].Value = 0U;

DmaChannelSpiTxTransferCfg0[5U].Param = Mcl_DmaSetDestination_TransferSize;

DmaChannelSpiTxTransferCfg0[5U].Value = DMA_IP_TRANSFER_SIZE_2_BYTE;

DmaChannelSpiTxTransferCfg0[6U].Param = Mcl_DmaSetMinorLoop_Size;

DmaChannelSpiTxTransferCfg0[6U].Value = 2U;

DmaChannelSpiTxTransferCfg0[7U].Param = Mcl_DmaSetMajorLoop_Count;

DmaChannelSpiTxTransferCfg0[7U].Value = 8U;

DmaChannelSpiTxTransferCfg0[8U].Param = Mcl_DmaSetControl_BandwidthControl;

DmaChannelSpiTxTransferCfg0[8U].Value = DMA_IP_BWC_ENGINE_NO_STALL;

Mcl_SetDmaChannelTransferList(CHANNEL_DSPI0_TX, &DmaChannelSpiTxTransferCfg0[0U], DMA_DSPI0_TX_TRANS

API: “void Mcl_SetDmaChannelScatterGatherList(const uint32 Channel, const uint32 Element, const Mcl_Dma↩ ChannelScatterGatherListType List[],const uint32 ListDimension);”

The Dma Channel can be configured during runtime, in case that the generated configuration by Tresos does not cover all the use cases. The Scatter Gather List configures Logic Channel Software TCD values, like: Transfer Control Settings, Source Settings, Destination Settings, MinorLoop Settings and MajorLoop Settings. The user shall create ScatterGather Configuration Lists (“Mcl_DmaChannelScatterGatherListType”) based on the required configuration. The “Element” is the Handler(Tag) generated by Tresos and identifies a Software TCD for the Logic Channel.

API: “void Mcl_GetDmaChannelParam(const uint32 Channel, const Mcl_DmaChannelInfoParameterType Param, uint32 ∗ const Value);”

The Dma Channel can return parameter information during runtime. The user selects the parameter from “Mcl_↩ DmaChannelInfoParameterType” enumeration. The value of the parameter shall be returned in the Value variable.

API: “void Mcl_SetDmaChannelScatterGatherConfig(const uint32 Channel, const uint32 Element);”

The Dma Channel can be configured during runtime, in case that the generated configuration by Tresos does not cover all the use cases. The Scatter Gather Configuration loads the selected Element for the Logic Channel into the Hardware TCD. The Element is internally configured to run in ScatterGather Mode.

6.3 Calls to Notification Functions, Callbacks, Callouts

calls to notification functions callbacks callouts template.

Chapter 7

Memory allocation

- Sections to be defined in _driver___MemMap.h
- Linker command file

7.1 Sections to be defined in _driver___MemMap.h

Section name	Type of section	Description
MCL_START_SEC_CODE	Code	Start of memory Section for Code
MCL_STOP_SEC_CODE	Code	End of memory Section for Code
MCL_START_SEC_CONFIG_DATA_UNSPECIFIED	Configuration Data	Start of Memory Section for Config Data
MCL_STOP_SEC_CONFIG_DATA_UNSPECIFIED	Configuration Data	End of Memory Section for Config Data
MCL_START_SEC_CONFIG_DATA_UNSPECIFIED_NO_CACHEABLE	Configuration Data	Start of Memory Section for Config Data (no cacheable).
MCL_STOP_SEC_CONFIG_DATA_UNSPECIFIED_NO_CACHEABLE	Configuration Data	End of Memory Section for Config Data (no cacheable).
MCL_START_SEC_VAR_CLEARED_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 cleared to zero by start-up code.
MCL_STOP_SEC_VAR_CLEARED_UNSPECIFIED	Variables	End of above section.
MCL_START_SEC_VAR_INIT_UNSPECIFIED	Variables	Used for variables, structures, arrays, when the SIZE (alignment) does not fit the criterian of 8,16 or 32 bit. These variables are initialized with values after every reset.
MCL_STOP_SEC_VAR_INIT_UNSPECIFIED	Variables	End of above section.
MCL_START_SEC_VAR_CLEARED_UNSPECIFIED_NO_CACHEABLE	Variables	Used for variables, structures, arrays when the SIZE (alignment) does not fit the criteria of 8,16 or 32 bit. These variables are cleared to zero by start-up code (no cacheable)
MCL_STOP_SEC_VAR_CLEARED_UNSPECIFIED_NO_CACHEABLE	Variables	End of above section.

7.2 Linker command file

Memory shall be allocated for every section defined in the driver's "<Module>"_MemMap.h.

Chapter 8

Integration Steps

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

1. Generate the required module configuration(s). For more details refer to section Files Required for Compilation
2. Allocate the proper memory sections in the driver's memory map header file ("<Module>"_MemMap.h) and linker command file. For more details refer to section Sections to be defined in <Module>_MemMap.h
3. Compile & build the module with all the dependent modules. For more details refer to section Building the Driver

Chapter 9

External assumptions for driver

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

External Assumption Req ID	External Assumption Text
EA_RTD_00071	If interrupts are locked, a centralized function pair to lock and unlock interrupts shall be used.
EA_RTD_00081	The integrator shall assure that <MSN>_Init() and <MSN>_DeInit() functions do not interrupt each other.
EA_RTD_00082	When caches are enabled and data buffers are allocated in cacheable memory regions the buffers involved in DMA transfer shall be aligned with both start and end to cache line size. Note: **Rationale**: This ensures that no other buffers/variables compete for the same cache lines.
EA_RTD_00106	Standalone IP configuration and HL configuration of the same driver shall be done in the same project
EA_RTD_00107	The integrator shall use the IP interface only for hardware resources that were configured for standalone IP usage. Note: The integrator shall not directly use the IP interface for hardware resources that were allocated to be used in HL context.
EA_RTD_00108	The integrator shall use the IP interface to a build a CDD, therefore the BSWMD will not contain reference to the IP interface

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.