Integration Manual

for S32R274 MCL Driver

Document Number: IM47MCLASR4.2 Rev002R1.0.0
Rev. 1.2

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

Section number Title Page

Chapter 6
Main API Requirements

Chapter 7
Memory Allocation

Chapter 8
Configuration parameters considerations

Chapter 9
Integration Steps

Chapter 10
External Assumptions for MCL driver

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0	25/11/2015	Livia Firan	RaceRunner Ultra 0.8.0 Release
1.1	05/02/2016	Khoa Dang	RaceRunner Ultra 0.9.0 Release
1.2	25/11/2016	Bach Nguyen	RaceRunner Ultra RTM 1.0.0 Release

Chapter 2
Introduction

This integration manual describes the integration requirements for MCL Driver for S32R274 microcontrollers.

2.1 Supported Derivatives

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

Table 2-1. S32R274 Derivatives

NXP	s32r274_mapbga257

All of the above microcontroller devices are collectively named as S32R274 .

2.2 Overview

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

AUTOSAR

- paves the way for innovative electronic systems that further improve performance, safety and environmental friendliness.
- is a strong global partnership that creates one common standard: "Cooperate on standards, compete on implementation".

- is a key enabling technology to manage the growing electrics/electronics complexity. It aims to be prepared for the upcoming technologies and to improve cost-efficiency without making any compromise with respect to quality.
- facilitates the exchange and update of software and hardware over the service life of the vehicle.

2.3 About this Manual

This Technical Reference employs the following typographical conventions:

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

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

Notes and warnings are shown as below:

Note

This is a note.

2.4 Acronyms and Definitions

Table 2-2. Acronyms and Definitions

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

2.5 Reference List

Table 2-3. Reference List

#	Title	Version
1	AUTOSAR 4.2 Rev002MCL Driver Software Specification Document.	4.2 Rev002
2	S32R274 Reference Manual	Rev. 2, RC, 10/2016
3	S32R274_1N58R Mask Set Errata (1N58R)	Rev.1

Chapter 3
Building the Driver

This section describes the source files and various compilers, linker options used for building the Autosar MCL driver for NXP S32R274 . It also explains the EB Tresos Studio plugin setup procedure.

3.1 Build Options

The MCL driver files are compiled using

- Windriver DIAB DIAB_5_9_4_8-FCS_20151130_213602
- Green Hills Multi 6.1.6 / Compiler 2015.1.4

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

Note

The TS_T2D47M10I0R0 plugin name is composed as follow:

TS_T = Target_Id

D = Derivative_Id

M = SW_Version_Major

I = SW_Version_Minor

R = Revision

(i.e. Target_Id = 2 identifies PowerPC architecture and Derivative_Id = 47 identifies the S32R274)

3.1.1 GHS Compiler/Linker/Assembler Options

Table 3-1. Compiler Options

Option	Description
-cpu=ppc5775kz7260	Selects target processor: ppc5775kz7260
-cpu=ppc5775kz4201	Selects target processor: ppc5775kz4201
-ansi	Specifies ANSI C with extensions. This mode extends the ANSI X3.159-1989 standard with certain useful and compatible constructs.
-noSPE	Disables the use of SPE and vector floating point instructions by the compiler.
-Ospace	Optimize for size.
-sda=0	Enables the Small Data Area optimization with a threshold of 0.
-vle	Enables VLE code generation
-dual_debug	Enables the generation of DWARF, COFF, or BSD debugging information in the object file
-G	Generates source level debugging information and allows procedure call from debugger's command line.
--no_exceptions	Disables support for exception handling
-Wundef	Generates warnings for undefined symbols in preprocessor expressions
-Wimplicit-int	Issues a warning if the return type of a function is not declared before it is called
-Wshadow	Issues a warning if the declaration of a local variable shadows the declaration of a variable of the same name declared at the global scope, or at an outer scope
-Wtrigraphs	Issues a warning for any use of trigraphs
--prototype_errors	Generates errors when functions referenced or called have no prototype
--incorrect_pragma_warnings	Valid #pragma directives with wrong syntax are treated as warnings
-noslashcomment	C++ like comments will generate a compilation error
-preprocess_assembly_files	Preprocesses assembly files
-nostartfile	Do not use Start files
--short_enum	Store enumerations in the smallest possible type
-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
-DUSE_SW_VECTOR_MODE	-D defines a preprocessor symbol and optionally can set it to a value. USE_SW_VECTOR_MODE: By default in the package, drivers are compiled to be used with interrupt controller configured to be in hardware vector mode. In case of AUTOSAR_OS_NOT_USED, the compiler option "-DUSE_SW_VECTOR_MODE" must be added to the list of compiler options to be used with interrupt controller configured to be in software vector mode.
-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.
-c	Produces an object file (called input-file.o) for each source file.

Table 3-2. Assembler Options

Option	Description
-cpu=ppc5775kz7260	Selects target processor: ppc5775kz7260
-cpu=ppc5775kz4201	Selects target processor: ppc5775kz4201

Table 3-3. Linker Options

Option	Description
-cpu=ppc5775kz7260	Selects target processor: ppc5775kz7260
-cpu=ppc5775kz4201	Selects target processor: ppc5775kz4201
-nostartfiles	Do not use Start files.
-vle	Enables VLE code generation

3.1.2 DIAB Compiler/Linker/Assembler Options

Table 3-4. Compiler Options

Option	Description
-tPPCE200Z4VEN:simple	Sets target processor to PPCE200Z4, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-tPPCE200Z7VEN:simple	Sets target processor to PPCE200Z7, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-Xdialect-ansi	Follow the ANSI C standard with some additions
-XO	Enables extra optimizations to produce highly optimized code
-g3	Generate symbolic debugger information and do all optimizations.
-Xsize-opt	Optimize for size rather than speed when there is a choice
-Xsmall-data=0	Set Size Limit for 'small data' Variables to zero.
-Xaddr-sconst=0x11	Specify addressing for constant static and global variables with size less than or equal to -Xsmall-const to far-absolute.
-Xaddr-sdata=0x11	Specify addressing for non-constant static and global variables with size less than or equal to -Xsmall-data in size to far-absolute.
-Xno-common	Disable use of the 'COMMON' feature so that the compiler or assembler will allocate each uninitialized public variable in the .bss section for the module defining it, and the linker will require exactly one definition of each public variable
-Xnested-interrupts	Allow nested interrupts
-Xdebug-dwarf2	Generate symbolic debug information in dwarf2 format
-Xdebug-local-all	Force generation of type information for all local variables
-Xdebug-local-cie	Create common information entry per module
-Xdebug-struct-all	Force generation of type information for all typedefs, struct, union and class types
-Xforce-declarations	Generates warnings if a function is used without a previous declaration

Table continues on the next page...

Table 3-4. Compiler Options (continued)

Option	Description
-ee1481	Generate an error when the function was used before it has been declared
-Xmacro-undefined-warn	Generates a warning when an undefined macro name occurs in a #if preprocessor directive
-Xlink-time-lint	Enable the checking of object and function declarations across compilation units, as well as the consistency of compiler options used to compile source files
-W:as:,-l	Pass the option '-l' (lower case letter L) to the assembler to get an assembler listing file
-Wa,-Xisa-vle	Instruct the assembler to expect and assemble VLE (Variable Length Encoding) instructions rather than BookE instructions.
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DUSE_SW_VECTOR_MODE	-D defines a preprocessor symbol and optionally can set it to a value. USE_SW_VECTOR_MODE: By default in the package, drivers are compiled to be used with interrupt controller configured to be in hardware vector mode. In case of AUTOSAR_OS_NOT_USED, the compiler option "-DUSE_SW_VECTOR_MODE" must be added to the list of compiler options to be used with interrupt controller configured to be in software vector mode.
-DDIAB	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the DIAB preprocessor symbol.
-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.
-c	Stop after assembly, produce object file.

Table 3-5. Assembler Options

Option	Description
-tPPCE200Z4VEN:simple	Sets target processor to PPCE200Z4, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-tPPCE200Z7VEN:simple	Sets target processor to PPCE200Z7, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-g	Dump the symbols in the global symbol table in each archive file.
-Xisa-vle	Expect and assemble VLE (Variable Length Encoding) instructions rather than Book E instructions. The default code section is named .text_vle instead of .text, and the default code section fill "character" is set to 0x44444444 instead of 0. The .text_vle code section will have ELF section header flags marking it as VLE code, not Book E code.
-Xasm-debug-on	Generate debug line and file information
-Xdebug-dwarf2	Generate symbolic debug information in dwarf2 format

Table 3-6. Linker Options

Option	Description
-tPPCE200Z4VEN:simple	Sets target processor to PPCE200Z4, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-tPPCE200Z7VEN:simple	Sets target processor to PPCE200Z7, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-Xelf	Generates ELF object format for output file
-m6	Generates a detailed link map and cross reference table
-lc	Specifies to linker to search for libc.a
-Xlibc-old	Enables usage of legacy (pre-release 5.6) libraries
-Xlink-time-lint	Enable the checking of object and function declarations across compilation units, as well as the consistency of compiler options used to compile source files

3.2 Files required for Compilation

This section describes the include files required to compile, assemble (if assembler code) and link the MCL driver for S32R274 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.

MCL Files

- ..\ MCL_TS_T2D47M10I0R0 \include\Mcl.h
- ..\ MCL_TS_T2D47M10I0R0 \include\CDD_Mcl.h
- ..\ MCL_TS_T2D47M10I0R0 \include\Mcl_Types.h
- ..\ MCL_TS_T2D47M10I0R0 \include\Mcl_EnvCfg.h
- ..\ MCL_TS_T2D47M10I0R0 \include\Mcl_Notif.h
- ..\ MCL_TS_T2D47M10I0R0 \include\Mcl_Dma.h
- ..\ MCL_TS_T2D47M10I0R0 \include\Mcl_Dma_Types.h
- ..\ MCL_TS_T2D47M10I0R0 \include\Mcl_DmaMux.h
- ..\ MCL_TS_T2D47M10I0R0 \include\Mcl_DmaMux_Types.h
- ..\ MCL_TS_T2D47M10I0R0 \include\Mcl_IPW.h
- ..\ MCL_TS_T2D47M10I0R0 \include\Mcl_IPW_Notif.h
- ..\ MCL_TS_T2D47M10I0R0 \include\Mcl_IPW_Types.h
- ..\ MCL_TS_T2D47M10I0R0 \include\Reg_eSys_Dma.h
- ..\ MCL_TS_T2D47M10I0R0 \include\Reg_eSys_DmaMux.h
- ..\ MCL_TS_T2D47M10I0R0 \include\eTimer_Common.h
- ..\ MCL_TS_T2D47M10I0R0 \include\eTimer_Common_Types.h

- ..\ MCL_TS_T2D47M10I0R0 \include\Reg_eSys_eTimer.h
- ..\ MCL_TS_T2D47M10I0R0 \include\Mcl_Axbs.h
- ..\ MCL_TS_T2D47M10I0R0 \include\Mcl_Axbs_Types.h
- ..\ MCL_TS_T2D47M10I0R0 \src\CDD_Mcl.c
- ..\ MCL_TS_T2D47M10I0R0 \src\Mcl_Dma.c
- ..\ MCL_TS_T2D47M10I0R0 \src\Mcl_Dma_Irq.c
- ..\ MCL_TS_T2D47M10I0R0 \src\Mcl_DmaMux.c
- ..\ MCL_TS_T2D47M10I0R0 \src\Mcl_IPW.c
- ..\ MCL_TS_T2D47M10I0R0 \src\eTimer_Common.c
- ..\ MCL_TS_T2D47M10I0R0 \src\Mcl_Axbs.c

MCL Generated Files

- CDD_Mcl_Cfg.c (For PC Variant) - For driver compilation, this file should be generated by the user using a configuration tool
- CDD_Mcl_[VariantName]_PBcfg.c - For driver compilation, this file should be generated by the user using a configuration tool.The file contains the definition of the init pointer for the respective variant.
- CDD_Mcl_Cfg.h - For driver compilation, this file should be generated by the user using a configuration tool
- As a deviation from standard:
  - CDD_Mcl_PBcfg_[VariantName].c files will contain the definition for all parameters that are variant aware, independent of the configuration class that will be selected (PC, LT, PB).
  - CDD_Mcl_Cfg.c file will contain the definition for all parameters that are not variant aware.

Files from Base common folder

- ..\Base_TS_T2D47M10I0R0 \include\Compiler.h
- ..\Base_TS_T2D47M10I0R0 \include\Compiler_Cfg.h
- ..\Base_TS_T2D47M10I0R0 \include\ComStack_Types.h
- ..\Base_TS_T2D47M10I0R0 \include\Mcl_MemMap.h
- ..\Base_TS_T2D47M10I0R0 \include\Mcal.h
- ..\Base_TS_T2D47M10I0R0 \include\Platform_Types.h
- ..\Base_TS_T2D47M10I0R0 \include\Std_Types.h
- ..\Base_TS_T2D47M10I0R0 \include\Reg_eSys.h
- ..\Base_TS_T2D47M10I0R0 \include\Soc_Ips.h
- ..\Base_TS_T2D47M10I0R0 \include\SilRegMacros.h

Files from Dem folder:

- ..\Dem_TS_T2D47M10I0R0 \include\Dem.h
- ..\Dem_TS_T2D47M10I0R0 \include\Dem_IntErrId.h
- ..\Dem_TS_T2D47M10I0R0 \include\Dem_Types.h

Files from Det folder:

- ..\Det_ TS_T2D47M10I0R0 \include\Det.h

3.3 Setting up the Plug-ins

The MCL driver was designed to be configured by using the EB Tresos Studio (version EB tresos Studio 21.0.0 b160607-0933 or later.)

Location of various files inside the MCL module folder:

- VSMD (Vendor Specific Module Definition) file in EB tresos Studio XDM format:
  - ..\ MCL _ TS_T2D47M10I0R0 \config\Mcl.xdm
  - ..\ Dem _ TS_T2D47M10I0R0 \config\Dem.xdm
  - ..\ Resource _ TS_T2D47M10I0R0 \config\Resource.xdm

- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - ..\ MCL _ TS_T2D47M10I0R0 \autosar\Mcl_<subderivative_name>.epd
  - ..\ Dem _ TS_T2D47M10I0R0 \autosar\Dem.epd
  - ..\ Resource _ TS_T2D47M10I0R0 \autosar \Resource_<subderivative_name>.epd

- Code Generation Templates for parameters without variation points:
  - ..\ MCL _ TS_T2D47M10I0R0 \output\src\CDD_Mcl_Cfg.c
  - ..\ MCL _ TS_T2D47M10I0R0 \output\include\CDD_Mcl_Cfg.h

- Code Generation Templates for variant aware parameters:
  - ..\ MCL _ TS_T2D47M10I0R0 \output\src\CDD_Mcl_PBCfg.c
  - ..\ MCL _ TS_T2D47M10I0R0 \output\include\CDD_Mcl_Cfg.h

Steps to generate the configuration:

1. Copy the module folders Mcl _ TS_T2D47M10I0R0 , Dem_ TS_T2D47M10I0R0 , Base_ TS_T2D47M10I0R0 , Resource_ TS_T2D47M10I0R0 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. Current Can driver has support for the following derivatives, everyone having attached a Resource file: s32r274_mapbga257 .

- **DET** is required for signaling the development error detection (parameters out of range, null pointers, etc).
- **DEM** is required for signaling the production error detection (hardware failure, etc).
- **ECUC** is required for configuring the variant handling in Tresos.

Chapter 4
Function calls to module

4.1 Function Calls during Start-up

The API to be called for this is Mcl _Init(). The MCU module should be initialized before MCL. All modules which use hardware initialized by MCL(eg DMA, AXBS) should be initialized after MCL.

4.2 Function Calls during Shutdown

The API to be called for this is Mcl_DeInit(). The MCU module should be deinitialized after MCL. All modules which use MCL features (example: DMA, Crossbar, Gtm, TrgMux) should be deinitialized before MCL.

4.3 Function Calls during Wake-up

NA.

Chapter 5
Module requirements

5.1 Exclusive areas to be defined in BSW scheduler

In the current implementation, MCL is using the services of Run-Time Environment (RTE) for entering and exiting the critical regions. RTE implementation is done by the integrators of the MCAL using OS or non-OS services. For testing the MCL, stubs are used for RTE. The following critical regions are used in the MCL driver:

5.1.1 MCL_EXCLUSIVE_AREA_00 With GTM, this area is used in function Gtm_TOM_EnableUpdateControl, protects the write to GTM_TOM_TGC0_GLB_CTRL register. With eMIOS, this area is used in function eMios_StartChannel, protects the write to EMIOSOUDIS register.

5.1.2 MCL_EXCLUSIVE_AREA_01 With GTM, this area is used in function Gtm_TOM_EnableUpdateControl, protects the write to GTM_TOM_TGC1_GLB_CTRL register. With eMIOS, this area is used in function eMios_StopChannel, protects the write to EMIOSOUDIS register.

5.1.3 MCL_EXCLUSIVE_AREA_02 With GTM, Used in function Gtm_TOM_DisableUpdateControl, protects the GTM_TOM_TGC0_GLB_CTRL register.

5.1.4 MCL_EXCLUSIVE_AREA_03 With GTM, Used in function Gtm_TOM_DisableUpdateControl, protects the GTM_TOM_TGC1_GLB_CTRL register.

5.1.5 MCL_EXCLUSIVE_AREA_04 With GTM, Used in function Gtm_TOM_EnableChannel, protects the GTM_TOM_TGC0_ENDIS_STAT register.

5.1.6 MCL_EXCLUSIVE_AREA_05 With GTM, Used in function Gtm_TOM_EnableChannel, protects the GTM_TOM_TGC1_ENDIS_STAT register.

5.1.7 MCL_EXCLUSIVE_AREA_06 With GTM, Used in function Gtm_TOM_DisableChannel, protects the GTM_TOM_TGC0_ENDIS_STAT register.

5.1.8 MCL_EXCLUSIVE_AREA_07 With GTM, Used in function Gtm_TOM_DisableChannel, protects the GTM_TOM_TGC1_ENDIS_STAT register.

5.1.9 MCL_EXCLUSIVE_AREA_08 With GTM, Used in function Gtm_TOM_EnableOutputChannel, protects the GTM_TOM_TGC0_OUTEN_STAT register.

5.1.10 MCL_EXCLUSIVE_AREA_09 With GTM, Used in function Gtm_TOM_EnableOutputChannel, protects the GTM_TOM_TGC1_OUTEN_STAT register.

5.1.11 MCL_EXCLUSIVE_AREA_10 With GTM, Used in function Gtm_TOM_DisableOutputChannel, protects the GTM_TOM_TGC0_OUTEN_STAT register.

5.1.1 MCL_EXCLUSIVE_AREA_11 With GTM, Used in function Gtm_TOM_DisableOutputChannel, protects the write to GTM_TOM_TGC1_OUTEN_STAT register.

5.1.2 MCL_EXCLUSIVE_AREA_12 With GTM, Used in function Dma_Init, protects the DMA_CR and DMA_CPR registers.

5.1.4 MCL_EXCLUSIVE_AREA_13 With DMA, used in function Dma_DeInit, protects the DMA_CR and DMA_CPR registers.

5.1.5 MCL_EXCLUSIVE_AREA_14 With DMA, Used in function Dma_SetChannelPriority, protects the DMA_CR and DMA_CPR registers.

5.1.6 MCL_EXCLUSIVE_AREA_15 With DMA, Used in function Dma_ConfigTcd, protects the TCDx.2ND_WORD, TCDx.6TH_WORD and TCDx.8TH_WORD registers.

5.1.7 MCL_EXCLUSIVE_AREA_16 With DMA, Used in function Mcl_IPW_DmaSetSModSize, protects the TCDx.2ND_WORD, TCDx.6TH_WORD and TCDx.8TH_WORD registers.

5.1.8 MCL_EXCLUSIVE_AREA_17 With DMA, Used in function Mcl_IPW_DmaSetDModSize, protects the TCDx.2ND_WORD, TCDx.6TH_WORD and TCDx.8TH_WORD registers.

5.1.9 MCL_EXCLUSIVE_AREA_18 With DMA, Used in function Mcl_IPW_DmaSetSoff, protects the TCDx.2ND_WORD, TCDx.6TH_WORD and TCDx.8TH_WORD registers.

5.1.10 MCL_EXCLUSIVE_AREA_19 With DMA, Used in function Mcl_IPW_DmaSetCiter, protects the TCDx.2ND_WORD, TCDx.6TH_WORD and TCDx.8TH_WORD registers.

5.1.11 MCL_EXCLUSIVE_AREA_20 With DMA, Used in function Mcl_IPW_DmaSetLinkAndIterCount, protects the TCDx.2ND_WORD, TCDx. 6TH_WORD and TCDx.8TH_WORD registers.

5.1.1 MCL_EXCLUSIVE_AREA_21 With DMA, Used in function Mcl_IPW_DmaSetDoff, protects the TCDx.2ND_WORD, TCDx.6TH_WORD and TCDx.8TH_WORD registers.

5.1.2 MCL_EXCLUSIVE_AREA_22 With DMA, Used in function Mcl_IPW_DmaSetIntMaj, protects the TCDx.2ND_WORD, TCDx.6TH_WORD and TCDx.8TH_WORD registers.

5.1.4 MCL_EXCLUSIVE_AREA_23 With DMA, Used in function Mcl_IPW_DmaClearIntMaj, protects the TCDx.2ND_WORD, TCDx.6TH_WORD and TCDx.8TH_WORD registers.

5.1.5 MCL_EXCLUSIVE_AREA_24 With DMA, Used in function Mcl_IPW_DmaSetFlags, protects the TCDx.2ND_WORD, TCDx.6TH_WORD and TCDx.8TH_WORD registers.

5.1.6 MCL_EXCLUSIVE_AREA_25 With DMA, Used in function Mcl_IPW_DmaSetBiter, protects the TCDx.2ND_WORD, TCDx.6TH_WORD and TCDx.8TH_WORD registers.

5.1.7 MCL_EXCLUSIVE_AREA_26 With DMA, Used in function Dma_ConfigLinkedTcd, protects the TCDx.2ND_WORD, TCDx.6TH_WORD and TCDx.8TH_WORD registers.

5.1.8 MCL_EXCLUSIVE_AREA_27 With DMA, Used in function Dma_ConfigScatterGatherTcd, protects the TCDx.2ND_WORD, TCDx.6TH_WORD and TCDx.8TH_WORD registers.

5.1.9 MCL_EXCLUSIVE_AREA_28 With DMA, Used in function Dma_DisableNotification, protects the TCDx.2ND_WORD, TCDx.6TH_WORD and TCDx.8TH_WORD registers.

5.1.10 MCL_EXCLUSIVE_AREA_29 With DMA, Used in function Dma_EnableNotification, protects the TCDx.2ND_WORD, TCDx.6TH_WORD and TCDx.8TH_WORD registers.

5.1.11 MCL_EXCLUSIVE_AREA_30 With DMA, Used in function Dma_ConfigScatterGatherChannel, protects the TCDx.2ND_WORD, TCDx. 6TH_WORD and TCDx.8TH_WORD registers.

5.1.1 MCL_EXCLUSIVE_AREA_31 With DMA, Used in function Dma_ConfigLinkedChannel protects the TCDx.2ND_WORD, TCDx.6TH_WORD and TCDx.8TH_WORD registers.

5.1.2 MCL_EXCLUSIVE_AREA_32 With LMem, Used in function Lmem_CacheDisablePc, protects the protects LMEM_PCCCR and LMEM_PSCCR registers (only on ARM platforms).

5.1.4 MCL_EXCLUSIVE_AREA_33 With LMem, Used in function Lmem_CacheDisablePs, protects the protects the protects LMEM_PCCCR and LMEM_PSCCR registers (only on ARM platforms).

5.1.5 MCL_EXCLUSIVE_AREA_34 With LMem, Used in function Lmem_CacheLaunchCommand, protects the protects LMEM_PCCCR and LMEM_PSCCR registers (only on ARM platforms).

Critical Region Exclusive Matrix

Below is the table depicting the exclusivity between different critical region IDs from the MCL 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

	MCL_EA_00	MCL_EA_01	MCL_EA_02	MCL_EA_03	MCL_EA_04	MCL_EA_05	MCL_EA_06	MCL_EA_07	MCL_EA_08	MCL_EA_09	MCL_EA_10	MCL_EA_11	MCL_EA_12	MCL_EA_13	MCL_EA_14	MCL_EA_15	MCL_EA_16	MCL_EA_17	MCL_EA_18	MCL_EA_19	MCL_EA_20	MCL_EA_21	MCL_EA_22	MCL_EA_23	MCL_EA_24	MCL_EA_25	MCL_EA_26	MCL_EA_27	MCL_EA_28	MCL_EA_29	MCL_EA_30	MCL_EA_31	MCL_EA_32	MCL_EA_33	MCL_EA_34
MCL_EA_00	x		x																																
MCL_EA_01		x		x																															

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

	MCL_EA_00	MCL_EA_01	MCL_EA_02	MCL_EA_03	MCL_EA_04	MCL_EA_05	MCL_EA_06	MCL_EA_07	MCL_EA_08	MCL_EA_09	MCL_EA_10	MCL_EA_11	MCL_EA_12	MCL_EA_13	MCL_EA_14	MCL_EA_15	MCL_EA_16	MCL_EA_17	MCL_EA_18	MCL_EA_19	MCL_EA_20	MCL_EA_21	MCL_EA_22	MCL_EA_23	MCL_EA_24	MCL_EA_25	MCL_EA_26	MCL_EA_27	MCL_EA_28	MCL_EA_29	MCL_EA_30	MCL_EA_31	MCL_EA_32	MCL_EA_33	MCL_EA_34
MCL_EA_02	x		x																																
MCL_EA_03		x		x																															
MCL_EA_04					x		x																												
MCL_EA_05						x		x																											
MCL_EA_06					x		x																												

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

	MCL_EA_00	MCL_EA_01	MCL_EA_02	MCL_EA_03	MCL_EA_04	MCL_EA_05	MCL_EA_06	MCL_EA_07	MCL_EA_08	MCL_EA_09	MCL_EA_10	MCL_EA_11	MCL_EA_12	MCL_EA_13	MCL_EA_14	MCL_EA_15	MCL_EA_16	MCL_EA_17	MCL_EA_18	MCL_EA_19	MCL_EA_20	MCL_EA_21	MCL_EA_22	MCL_EA_23	MCL_EA_24	MCL_EA_25	MCL_EA_26	MCL_EA_27	MCL_EA_28	MCL_EA_29	MCL_EA_30	MCL_EA_31	MCL_EA_32	MCL_EA_33	MCL_EA_34
MCL_EA_07						x		x																											
MCL_EA_08									x		x																								
MCL_EA_09										x		x																							
MCL_EA_10									x		x																								
MCL_EA_11										x		x																							

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

	MCL_EA_00	MCL_EA_01	MCL_EA_02	MCL_EA_03	MCL_EA_04	MCL_EA_05	MCL_EA_06	MCL_EA_07	MCL_EA_08	MCL_EA_09	MCL_EA_10	MCL_EA_11	MCL_EA_12	MCL_EA_13	MCL_EA_14	MCL_EA_15	MCL_EA_16	MCL_EA_17	MCL_EA_18	MCL_EA_19	MCL_EA_20	MCL_EA_21	MCL_EA_22	MCL_EA_23	MCL_EA_24	MCL_EA_25	MCL_EA_26	MCL_EA_27	MCL_EA_28	MCL_EA_29	MCL_EA_30	MCL_EA_31	MCL_EA_32	MCL_EA_33	MCL_EA_34
MCL_EA_12															x																				
MCL_EA_13															x																				
MCL_EA_14													x	x	x																				
MCL_EA_15																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			
MCL_EA_16																x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

	MCL_EA_00	MCL_EA_01	MCL_EA_02	MCL_EA_03	MCL_EA_04	MCL_EA_05	MCL_EA_06	MCL_EA_07	MCL_EA_08	MCL_EA_09	MCL_EA_10	MCL_EA_11	MCL_EA_12	MCL_EA_13	MCL_EA_14	MCL_EA_15	MCL_EA_16	MCL_EA_17	MCL_EA_18	MCL_EA_19	MCL_EA_20	MCL_EA_21	MCL_EA_22	MCL_EA_23	MCL_EA_24	MCL_EA_25	MCL_EA_26	MCL_EA_27	MCL_EA_28	MCL_EA_29	MCL_EA_30	MCL_EA_31	MCL_EA_32	MCL_EA_33	MCL_EA_34
MCL_EA_17																x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			
MCL_EA_18																x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			
MCL_EA_19																x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			
MCL_EA_20																x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			
MCL_EA_21																x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

	MCL_EA_00	MCL_EA_01	MCL_EA_02	MCL_EA_03	MCL_EA_04	MCL_EA_05	MCL_EA_06	MCL_EA_07	MCL_EA_08	MCL_EA_09	MCL_EA_10	MCL_EA_11	MCL_EA_12	MCL_EA_13	MCL_EA_14	MCL_EA_15	MCL_EA_16	MCL_EA_17	MCL_EA_18	MCL_EA_19	MCL_EA_20	MCL_EA_21	MCL_EA_22	MCL_EA_23	MCL_EA_24	MCL_EA_25	MCL_EA_26	MCL_EA_27	MCL_EA_28	MCL_EA_29	MCL_EA_30	MCL_EA_31	MCL_EA_32	MCL_EA_33	MCL_EA_34
MCL_EA_22																x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			
MCL_EA_23																x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			
MCL_EA_24																x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			
MCL_EA_25																x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			
MCL_EA_26																x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

	MCL_EA_00	MCL_EA_01	MCL_EA_02	MCL_EA_03	MCL_EA_04	MCL_EA_05	MCL_EA_06	MCL_EA_07	MCL_EA_08	MCL_EA_09	MCL_EA_10	MCL_EA_11	MCL_EA_12	MCL_EA_13	MCL_EA_14	MCL_EA_15	MCL_EA_16	MCL_EA_17	MCL_EA_18	MCL_EA_19	MCL_EA_20	MCL_EA_21	MCL_EA_22	MCL_EA_23	MCL_EA_24	MCL_EA_25	MCL_EA_26	MCL_EA_27	MCL_EA_28	MCL_EA_29	MCL_EA_30	MCL_EA_31	MCL_EA_32	MCL_EA_33	MCL_EA_34
MCL_EA_27																x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			
MCL_EA_28																x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			
MCL_EA_29																x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			
MCL_EA_30																x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			
MCL_EA_31																x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

	MCL_EA_00	MCL_EA_01	MCL_EA_02	MCL_EA_03	MCL_EA_04	MCL_EA_05	MCL_EA_06	MCL_EA_07	MCL_EA_08	MCL_EA_09	MCL_EA_10	MCL_EA_11	MCL_EA_12	MCL_EA_13	MCL_EA_14	MCL_EA_15	MCL_EA_16	MCL_EA_17	MCL_EA_18	MCL_EA_19	MCL_EA_20	MCL_EA_21	MCL_EA_22	MCL_EA_23	MCL_EA_24	MCL_EA_25	MCL_EA_26	MCL_EA_27	MCL_EA_28	MCL_EA_29	MCL_EA_30	MCL_EA_31	MCL_EA_32	MCL_EA_33	MCL_EA_34
MCL_EA_32																																	x	x	x
MCL_EA_33																																	x	x	x
MCL_EA_34																																	x	x	x

Note

- **MCL_EA_xx** means **MCL_EXCLUSIVE_AREA_xx**

5.2 Peripheral Hardware Requirements

None.

5.3 ISR to configure within OS – dependencies

The following ISR’s are used by the MCL driver:

The ISR table is presented below. Depending on the derivative used, some of the ISRs may not be available. For complete details please consult the Reference Manual:

Table 5-2. eDMA 0 interrupts

eDMA 0 Interrupts	Hardware interrupt vector
MCL_DMA_ERROR_ISR	52
MCL_DMA_CH_0_ISR	53
MCL_DMA_CH_1_ISR	54
MCL_DMA_CH_2_ISR	55
MCL_DMA_CH_3_ISR	56
MCL_DMA_CH_4_ISR	57
MCL_DMA_CH_5_ISR	58
MCL_DMA_CH_6_ISR	59
MCL_DMA_CH_7_ISR	60
MCL_DMA_CH_8_ISR	61
MCL_DMA_CH_9_ISR	62
MCL_DMA_CH_10_ISR	63
MCL_DMA_CH_11_ISR	64
MCL_DMA_CH_12_ISR	65
MCL_DMA_CH_13_ISR	66
MCL_DMA_CH_14_ISR	67
MCL_DMA_CH_15_ISR	68
MCL_DMA_CH_16_ISR	69
MCL_DMA_CH_17_ISR	70
MCL_DMA_CH_18_ISR	71
MCL_DMA_CH_19_ISR	72
MCL_DMA_CH_20_ISR	73
MCL_DMA_CH_21_ISR	74
MCL_DMA_CH_22_ISR	75
MCL_DMA_CH_23_ISR	76
MCL_DMA_CH_24_ISR	77
MCL_DMA_CH_25_ISR	78
MCL_DMA_CH_26_ISR	79
MCL_DMA_CH_27_ISR	80
MCL_DMA_CH_28_ISR	81
MCL_DMA_CH_29_ISR	82
MCL_DMA_CH_30_ISR	83
MCL_DMA_CH_31_ISR	84
ETIMER_1_CH_0_ISR	622
ETIMER_1_CH_1_ISR	623
ETIMER_1_CH_2_ISR	624
ETIMER_1_CH_3_ISR	625

Table continues on the next page...

Table 5-2. eDMA 0 interrupts (continued)

eDMA 0 Interrupts	Hardware interrupt vector
ETIMER_1_CH_4_ISR	626
ETIMER_1_CH_5_ISR	627
ETIMER_1_CH_6_ISR	628
ETIMER_1_CH_7_ISR	632
ETIMER_2_CH_0_ISR	633
ETIMER_2_CH_1_ISR	634
ETIMER_2_CH_2_ISR	635
ETIMER_2_CH_3_ISR	636
ETIMER_2_CH_4_ISR	637
ETIMER_2_CH_5_ISR	638
ETIMER_2_CH_5_ISR	639
ETIMER_2_CH_7_ISR	640
ETIMER_2_CH_8_ISR	643

NOTE

In case of AUTOSAR_OS_NOT_USED, the compiler option "-DUSE_HW_VECTOR_MODE" must be added to the list of compiler options to be used with interrupt controller configured to be in hardware vector mode.

5.4 ISR Macro

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

a. OS is not used - AUTOSAR_OS_NOT_USED is defined:

i. If USE_SW_VECTOR_MODE is defined:

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

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

ii. If USE_SW_VECTOR_MODE is not defined:

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

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

Custom OS is used - AUTOSAR_OS_NOT_USED is not defined

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

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

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

5.5 Other AUTOSAR modules - dependencies

- **Det** This module is necessary for enabling Development error detection. The API function used is Det_ReportError(). The activation/deactivation of Development error detection is configurable using 'CanDevErrorDetect' configuration parameter.
- **Dem:** This module is necessary for enabling reporting of production relevant error status. The API function used is Dem_ReportErrorStatus().
- **Resource:** Sub-Derivative model is selected from Resource configuration.
- **EcuC:** Is required for configuring the variant handling in Tresos.

5.6 Data Cache Restriction

DMA trasfers may issue cache coherency problems. To avoid possible coherency issues when D-CACHE is enabled, the integrator shall ensure that the buffers used as TCD source and destination are allocated in the NON-CACHEABLE area(by means of Memmap).

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

Call-back Notifications:

None.

User Notification:

The MCL Driver provides a notification per channel. The ISRs shall be responsible for resetting the interrupt's flags (if needed by hardware) and calling the corresponding notification function. The notifications can be configured as pointers to user defined functions. If notification is not desired, NULL_PTR shall be configured.

Mcl_Notification_<Channel>

The syntax of this function is as follows:

void NotificationName

(

void

)

According to the last call of Mcl_EnableNotification, this notification function shall be called when the major iteration count completes or when the major iteration is half complete.

Chapter 7
Memory Allocation

7.1 Sections to be defined in MemMap.h

Tables descibe Sections to be defined in MemMap.h:

Table 7-1. Sectionto be define

<Section name>	Tyep of section	Description
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_CONST_UNSPECIFIED	Configuration Data	Used for configuration data which is not variant aware.
MCL_START_SEC_CODE	Code	Start of memory Section for Code in flash.
MCL_STOP_SEC_CODE	Code	Stop of memory Section for Code in flash.
MCL_START_SEC_RAMCODE	Code	Start of memory Section for Code in ram.
MCL_STOP_SEC_RAMCODE	Code	Stop of memory Section for Code in ram.
MCL_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.
MCL_STOP_SEC_VAR_INIT_UNSPECIFIED	Variables	End of above section.

Table continues on the next page...

Table 7-1. Sectionto be define (continued)

MCL_START_SEC_VAR_INIT_16	Variables	Used for variables which have to be aligned to 16 bit. For instance used for variables of size 16 bit or used for composite data types: arrays, structs containing elements of maximum 16 bits. These variables are initialized with values after every reset
MCL_STOP_SEC_VAR_INIT_16	Variables	End of above section.
MCL_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 (BBS).
MCL_STOP_SEC_VAR_NO_INIT_UNSPECIFIED	Variables	End of above section.

7.2 Linker command file

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

Chapter 8
Configuration parameters considerations

Configuration parameter class for Autosar MCL 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
Mcl	IMPLEMENTATION_CONFIG_VARIANT	Pre Compile parameter for all Variants of Configuration	Pre Compile
DmaInstance	MclEDMA_CX	VariantPC or VariantPB	Post Build
	MclEDMA_ECX	VariantPC or VariantPB	Post Build
	MclEDMAChGroup1Priority	VariantPC or VariantPB	Post Build
	MclEDMAChGroup0Priority	VariantPC or VariantPB	Post Build
	MclEDMA_CLM	VariantPC or VariantPB	Post Build
	MclEDMA_HALT	VariantPC or VariantPB	Post Build
	MclEDMA_HOE	VariantPC or VariantPB	Post Build
	MclEDMA_ERGA	VariantPC or VariantPB	Post Build
	MclEDMA_ERCA	VariantPC or VariantPB	Post Build
	MclEDMA_EDBG	VariantPC or VariantPB	Post Build
DMAChannel	DMAChannelPriority	VariantPC or VariantPB	Post Build
	ECP	VariantPC or VariantPB	Post Build
	DPA	VariantPC or VariantPB	Post Build
	EMI	VariantPC or VariantPB	Post Build
	MclDMAChannelId	VariantPC or VariantPB	Post Build
	DmaHwChannel	VariantPC or VariantPB	Post Build
	MclDmaTransferCompletionNotif	VariantPC or VariantPB	Post Build
	MclDMAChannelEnable	VariantPC or VariantPB	Post Build

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
	MclDMAChannelTriggerEnable	VariantPC or VariantPB	Post Build
	DmaSource0	VariantPC or VariantPB	Post Build
	DmaSource1	VariantPC or VariantPB	Post Build
MclGeneral	MclDisableDemReportErrorStatus	Pre Compile parameter for all Variants of Configuration	Pre Compile
	MclDevErrorDetect	VariantPC or VariantPB	Post Build
	MclErrorChecking	VariantPC or VariantPB	Post Build
	Mcl_VersionInfoApi	VariantPC or VariantPB	Post Build
	Mcl_DmaGetChannelErrorStatusApi	VariantPC or VariantPB	Post Build
	Mcl_DmaGetGlobalErrorStatusApi	VariantPC or VariantPB	Post Build
	EnableDMA	VariantPC or VariantPB	Post Build
	MclEnableCrossbarSwitch	VariantPC or VariantPB	Post Build
	MclErrorNotificationDma0	VariantPC or VariantPB	Post Build
	Mcl_DeInitApi	VariantPC or VariantPB	Post Build
	MclEnableCrossbarSwitch	VariantPC or VariantPB	Post Build
MclDemEventParameterRefs	MCL_DMA_E_DESCRIPTOR	Pre Compile parameter for all Variants of Configuration	Pre Compile
	MCL_DMA_E_ECC	Pre Compile parameter for all Variants of Configuration	Pre Compile
	MCL_DMA_E_BUS	Pre Compile parameter for all Variants of Configuration	Pre Compile
	MCL_DMA_E_PRIORITY	Pre Compile parameter for all Variants of Configuration	Pre Compile
	MCL_DMA_E_INCONSISTENCY	Pre Compile parameter for all Variants of Configuration	Pre Compile
	MCL_DMA_E_UNRECOGNIZED	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

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
	SwPatchVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	VendorApiInfix	Pre Compile parameter for all Variants of Configuration	Pre Compile
	VendorId	Pre Compile parameter for all Variants of Configuration	Pre Compile
MclIsrAvailable	MclIsrName	VariantPC or VariantPB	Post Build
	MclIsrEnabled	VariantPC or VariantPB	Post Build
MclCrossbarLogicalSlavePorts	MclCrossbarLogicalSlavePortId	VariantPC or VariantPB	Post Build
	MclCrossbarLogicalSlavePortRef	VariantPC or VariantPB	Post Build
MclCrossbarLogicalMasterPorts	MclCrossbarLogicalMasterPortId	VariantPC or VariantPB	Post Build
	MclCrossbarLogicalMasterPortRef	VariantPC or VariantPB	Post Build
MclCrossbarInstance	MclCrossbarHwInstance	VariantPC or VariantPB	Post Build
MclCrossbarInstance/ MclCrossbarHwSlavePort	MclSlavePortNumber	VariantPC or VariantPB	Post Build
	MclCrossbarPrioMaster0	VariantPC or VariantPB	Post Build
	MclCrossbarPrioMaster1	VariantPC or VariantPB	Post Build
	MclCrossbarPrioMaster2	VariantPC or VariantPB	Post Build
	MclCrossbarPrioMaster3	VariantPC or VariantPB	Post Build
	MclCrossbarPrioMaster4	VariantPC or VariantPB	Post Build
	MclCrossbarPrioMaster5	VariantPC or VariantPB	Post Build
	MclCrossbarPrioMaster6	VariantPC or VariantPB	Post Build
	MclCrossbarPrioMaster7	VariantPC or VariantPB	Post Build
	MclCrossbarEnableLock	VariantPC or VariantPB	Post Build
	MclCrossbarHaltLowPrio	VariantPC or VariantPB	Post Build
	MclCrossbarEnablePrioElevM0	VariantPC or VariantPB	Post Build
	MclCrossbarEnablePrioElevM1	VariantPC or VariantPB	Post Build
	MclCrossbarEnablePrioElevM2	VariantPC or VariantPB	Post Build
	MclCrossbarEnablePrioElevM3	VariantPC or VariantPB	Post Build
	MclCrossbarEnablePrioElevM4	VariantPC or VariantPB	Post Build
	MclCrossbarEnablePrioElevM5	VariantPC or VariantPB	Post Build
	MclCrossbarEnablePrioElevM6	VariantPC or VariantPB	Post Build
	MclCrossbarEnablePrioElevM7	VariantPC or VariantPB	Post Build

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
	MclCrossbarEnableFixedPrio	VariantPC or VariantPB	Post Build
	MclCrossbarParkingControl	VariantPC or VariantPB	Post Build
	MclCrossbarParkField	VariantPC or VariantPB	Post Build
MclCrossbarInstance/ MclCrossbarHwMasterPort	MclMasterPortNumber	VariantPC or VariantPB	Post Build
	MclCrossbarArbitrates	VariantPC or VariantPB	Post Build

Chapter 9
Integration Steps

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

- Generate the required MCL configurations. For more details refer to section Files required for Compilation
- Allocate proper memory sections in MCL_MemMap.h and linker command file. For more details refer to section
- Compile & build the MCL with all the dependent modules. For more details refer to section Building the Driver

Chapter 10
External Assumptions for MCL driver

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

[SMCAL_CPR_EXT163]

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

[SMCAL_CPR_EXT176]

<< The integrator shall assure that (MSN)_Init() and (MSN)_DeInit() functions do not interrupt each other. >>

[SMCAL_CPR_EXT177]

<< When caches are enabled and data buffers are allocated in cachable 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 to compete for the same cache lines.

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.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C-5, CodeTest, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and μVision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

Document Number IM47MCLASR4.2 Rev002R1.0.0
Revision 1.2