

---

# User Manual

for S32R274 MCL Driver

Document Number: UM47MCLASR4.2 Rev002R1.0.0  
Rev. 1.2





# Contents

Section number	Title	Page
<b>Chapter 1</b>		
<b>Revision History</b>		
<b>Chapter 2</b>		
<b>Introduction</b>		
2.1	Supported Derivatives.....	15
2.2	Overview.....	15
2.3	About this Manual.....	16
2.4	Acronyms and Definitions.....	16
2.5	Reference List.....	16
<b>Chapter 3</b>		
<b>Driver</b>		
3.1	Requirements.....	19
3.2	Driver Design Summary.....	19
3.3	Hardware Resources.....	19
3.4	Deviation from requirements.....	20
3.5	Driver Limitations.....	20
3.6	Driver Usage and Configuration tips.....	20
3.7	Runtime Errors.....	21
3.8	Software specification.....	25
3.8.1	Define Reference.....	25
3.8.1.1	Define MCL_ACKNOWLEDGEINTERRUPT_ID_U8.....	25
3.8.1.2	Define MCL_AR_RELEASE_MAJOR_VERSION.....	25
3.8.1.3	Define MCL_AR_RELEASE_MINOR_VERSION.....	26
3.8.1.4	Define MCL_AR_RELEASE_REVISION_VERSION.....	26
3.8.1.5	Define MCL_CONFIG_CH_ID_U8.....	26
3.8.1.6	Define MCL_CONFIG_LINK_CH_ID_U8.....	27
3.8.1.7	Define MCL_CONFIG_LINK_TCD_ID_U8.....	27
3.8.1.8	Define MCL_CONFIG_SCA_CH_ID_U8.....	27

Section number	Title	Page
3.8.1.9	Define MCL_CONFIG_SCA_LINK_CH_ID_U8.....	28
3.8.1.10	Define MCL_CONFIG_SCA_LINK_TCD_ID_U8.....	28
3.8.1.11	Define MCL_CONFIG_SCA_TCD_ID_U8.....	28
3.8.1.12	Define MCL_CONFIG_TCD_ID_U8.....	29
3.8.1.13	Define MCL_DISABLENOTIFICATION_ID_U8.....	29
3.8.1.14	Define MCL_DMACLEARDONE_ID_U8.....	29
3.8.1.15	Define MCL_DMAGETCHANNELTCDADDRESS_ID_U8.....	30
3.8.1.16	Define MCL_DMAGETINTERRUPTREQUEST_ID_U8.....	30
3.8.1.17	Define MCL_DMAGETPHYSICALCHANNEL_ID_U8.....	31
3.8.1.18	Define MCL_DMATCDGETINTMAJ_ID_U8.....	31
3.8.1.19	Define MCL_E_ALREADY_INITIALIZED_U8.....	31
3.8.1.20	Define MCL_E_INVALID_CHANNEL_U8.....	32
3.8.1.21	Define MCL_E_PARAM_CONFIG_U8.....	32
3.8.1.22	Define MCL_E_PARAM_NOTIFICATION_NULL_U8.....	32
3.8.1.23	Define MCL_E_PARAM_POINTER_U8.....	33
3.8.1.24	Define MCL_E_PARAM_VINFO_U8.....	33
3.8.1.25	Define MCL_E_UNEXPECTED_ISR_U8.....	33
3.8.1.26	Define MCL_E_UNINIT_U8.....	34
3.8.1.27	Define MCL_ENABLENOTIFICATION_ID_U8.....	34
3.8.1.28	Define MCL_GETVERSIONINFO_ID_U8.....	34
3.8.1.29	Define MCL_GET_GLOBAL_ERR_STATUS_ID_U8.....	35
3.8.1.30	Define MCL_GET_CH_ERR_STATUS_ID_U8.....	35
3.8.1.31	Define MCL_DMA_NO_CHANNEL_U16.....	36
3.8.1.32	Define MCL_DMA_CHANNEL_NOT_CONFIGURED_U8.....	36
3.8.1.33	Define MCL_INIT_ID_U8.....	36
3.8.1.34	Define MCL_MODULE_ID.....	36
3.8.1.35	Define MCL_SET_PRI_ID_U8.....	37
3.8.1.36	Define MCL_START_CH_ID_U8.....	37
3.8.1.37	Define MCL_SW_MAJOR_VERSION.....	37

Section number	Title	Page
3.8.1.38	Define MCL_SW_MINOR_VERSION.....	38
3.8.1.39	Define MCL_SW_PATCH_VERSION.....	38
3.8.1.40	Define MCL_TRANSF_ACTIVE_ID_U8.....	38
3.8.1.41	Define MCL_TRANSF_COMPL_ID_U8.....	39
3.8.1.42	Define MCL_VENDOR_ID.....	39
3.8.1.43	Define DMA_TCD_DONE_U8.....	39
3.8.1.44	Define DMA_TCD_ACTIVE_U8.....	39
3.8.1.45	Define DMA_TCD_MAJOR_E_LINK_U8.....	40
3.8.1.46	Define DMA_TCD_E_SG_U8.....	40
3.8.1.47	Define DMA_TCD_DISABLE_REQ_U8.....	40
3.8.1.48	Define DMA_TCD_INT_HALF_U8.....	40
3.8.1.49	Define DMA_TCD_INT_MAJOR_U8.....	41
3.8.1.50	Define DMA_TCD_START_U8.....	41
3.8.1.51	Define MCL_GET_CRT_ITER_CH_ID_U8.....	41
3.8.1.52	Define MCL_GET_STRT_ITER_CH_ID_U8.....	41
3.8.1.53	Define MCL_UPDATE_DEST_CH_ID_U8.....	42
3.8.1.54	Define MCL_UPDATE_ITER_ID_U8.....	42
3.8.2	Enum Reference.....	42
3.8.2.1	Enumeration Mcl_DmaTransferNotifType.....	42
3.8.2.2	Enumeration Mcl_DmaSizeType.....	43
3.8.2.3	Enumeration Mcl_DmaRequestType.....	43
3.8.2.4	Enumeration Mcl_DmaChannelErrorStatusType.....	44
3.8.3	Function Reference.....	44
3.8.3.1	Function Mcl_DmaAcknowledgeInterrupt.....	44
3.8.3.2	Function Mcl_DmaConfigChannel.....	45
3.8.3.3	Function Mcl_DmaConfigLinkedChannel.....	46
3.8.3.4	Function Mcl_DmaConfigLinkedTcd.....	46
3.8.3.5	Function Mcl_DmaConfigScatterGatherChannel.....	47
3.8.3.6	Function Mcl_DmaConfigScatterGatherLinkedChannel.....	48

Section number	Title	Page
3.8.3.7	Function Mcl_DmaConfigScatterGatherLinkedTcd.....	48
3.8.3.8	Function Mcl_DmaConfigScatterGatherTcd.....	49
3.8.3.9	Function Mcl_DmaConfigTcd.....	50
3.8.3.10	Function Mcl_DmaEnableNotification.....	50
3.8.3.11	Function Mcl_DmaDisableNotification.....	51
3.8.3.12	Function Mcl_DmaEnableHwRequest.....	51
3.8.3.13	Function Mcl_DmaDisableHwRequest.....	52
3.8.3.14	Function Mcl_DmaGetChannelTcdAddress.....	52
3.8.3.15	Function Mcl_DmaGetInterruptRequest.....	53
3.8.3.16	Function Mcl_DmaGetPhysicalChannel.....	53
3.8.3.17	Function Mcl_DmaIsTransferActive.....	54
3.8.3.18	Function Mcl_DmaIsTransferCompleted.....	54
3.8.3.19	Function Mcl_DmaSetChannelPriority.....	55
3.8.3.20	Function Mcl_DmaStartChannel.....	56
3.8.3.21	Function Mcl_DmaTcdClearDone.....	56
3.8.3.22	Function Mcl_DmaTcdClearIntMaj.....	57
3.8.3.23	Function Mcl_DmaTcdGetDaddr.....	57
3.8.3.24	Function Mcl_DmaTcdGetFlags.....	58
3.8.3.25	Function Mcl_DmaTcdGetIntMaj.....	58
3.8.3.26	Function Mcl_DmaTcdGetIterCount.....	59
3.8.3.27	Function Mcl_DmaTcdGetSaddr.....	59
3.8.3.28	Function Mcl_DmaTcdSetDaddr.....	60
3.8.3.29	Function Mcl_DmaTcdSetDlast.....	60
3.8.3.30	Function Mcl_DmaTcdSetDModuloAndSize.....	61
3.8.3.31	Function Mcl_DmaTcdSetDoff.....	61
3.8.3.32	Function Mcl_DmaTcdSetFlags.....	62
3.8.3.33	Function Mcl_DmaTcdSetIntMaj.....	63
3.8.3.34	Function Mcl_DmaTcdSetIterCount.....	63
3.8.3.35	Function Mcl_DmaTcdSetLinkAndIterCount.....	64

Section number	Title	Page
3.8.3.36	Function Mcl_DmaTcdSetMinorLoop.....	64
3.8.3.37	Function Mcl_DmaTcdSetSaddr.....	65
3.8.3.38	Function Mcl_DmaTcdSetSga.....	66
3.8.3.39	Function Mcl_DmaTcdSetSlast.....	66
3.8.3.40	Function Mcl_DmaTcdSetSModuloAndSize.....	67
3.8.3.41	Function Mcl_DmaTcdSetSoff.....	67
3.8.3.42	Function Mcl_DmaUpdateDestAddress.....	68
3.8.3.43	Function Mcl_DmaUpdateIterCount.....	68
3.8.3.44	Function Mcl_DmaGetCrtIterCount.....	69
3.8.3.45	Function Mcl_DmaGetStartIterCount.....	69
3.8.3.46	Function Mcl_Init.....	70
3.8.3.47	Function Mcl_DeInit.....	70
3.8.3.48	Function Mcl_DmaGetGlobalErrorStatus.....	71
3.8.3.49	Function Mcl_DmaGetChannelErrorStatus.....	72
3.8.3.50	Function Mcl_GetVersionInfo.....	72
3.8.4	Structs Reference.....	73
3.8.4.1	Structure Mcl_AxbsPortConfigType.....	73
3.8.4.2	Structure Mcl_ChannelConfigType.....	74
3.8.4.3	Structure Mcl_CrossbarConfigType.....	74
3.8.4.4	Structure Mcl_DmaInitConfigType.....	75
3.8.4.5	Structure Mcl_ConfigType .....	77
3.8.4.6	Structure Mcl_DmaChannelConfigType.....	78
3.8.4.7	Structure Mcl_DmaConfigType.....	79
3.8.4.8	Structure Mcl_DmaMuxChannelConfigType.....	80
3.8.4.9	Structure Mcl_DmaMuxConfigType.....	80
3.8.4.10	Structure Mcl_DmaTcdAttributesType.....	81
3.8.4.11	Structure Mcl_DmaHwIpsConfigType.....	83
3.8.4.12	Structure Mcl_DmaGlobalErrorStatusType.....	84
3.8.5	Types Reference.....	84

Section number	Title	Page
3.8.5.1	Typedef Mcl_DmaChannelType.....	85
3.8.5.2	Typedef Mcl_DmaControlType.....	85
3.8.5.3	Typedef Mcl_DmaPriorityType.....	85
3.8.5.4	Typedef Mcl_DmaTcdType.....	85
3.8.5.5	Typedef Mcl_DmaMuxChannelType.....	85
3.8.5.6	Typedef Mcl_ChannelType.....	86
3.8.5.7	Typedef Mcl_NotifyType.....	86
3.8.5.8	Typedef Mcl_DmaInstanceType.....	87
3.8.5.9	Typedef Mcl_DmaErroneousChannelType.....	87
3.8.5.10	Typedef Axbs_InstanceNumber.....	87
3.8.5.11	Typedef Mcl_AxbsPortControlType.....	87
3.8.5.12	Typedef Mcl_AxbsPortNumberType.....	88
3.8.5.13	Typedef Mcl_AxbsPortPriorityType.....	88
3.8.5.14	Typedef Mcl_CrossbarHwIpsConfigType.....	88
3.8.5.15	Typedef Mcl_CrossbarPortType.....	88

## Chapter 4 Tresos Configuration Plug-in

4.1	Configuration elements of Mcl.....	89
4.2	Form IMPLEMENTATION_CONFIG_VARIANT.....	89
4.3	Form MclGeneral.....	90
4.3.1	MclDisableDemReportErrorStatus (MclGeneral).....	90
4.3.2	MclDevErrorDetect (MclGeneral).....	90
4.3.3	MclErrorChecking (MclGeneral).....	91
4.3.4	Mcl_VersionInfoApi (MclGeneral).....	91
4.3.5	Mcl_DmaGetChannelErrorStatusApi(MclGeneral).....	91
4.3.6	Mcl_DmaGetGlobalErrorStatusApi(MclGeneral).....	92
4.3.7	EnableDMA (MclGeneral).....	92
4.3.8	MclEnableCrossbarSwitch (MclGeneral).....	92
4.3.9	MclErrorNotificationDma0 (MclGeneral).....	93



Section number	Title	Page
4.4	Form McIDemEventParameterRefs.....	93
4.4.1	MCL_DMA_E_DESCRIPTOR (McIDemEventParameterRefs).....	94
4.4.2	MCL_DMA_E_ECC (McIDemEventParameterRefs).....	94
4.4.3	MCL_DMA_E_BUS (McIDemEventParameterRefs).....	95
4.4.4	MCL_DMA_E_PRIORITY (McIDemEventParameterRefs).....	95
4.4.5	MCL_DMA_E_INCONSISTENCY (McIDemEventParameterRefs).....	96
4.4.6	MCL_DMA_E_UNRECOGNIZED (McIDemEventParameterRefs).....	96
4.5	Form McIConfigSet.....	97
4.5.1	McIEDMA_CX (McIConfigSet).....	97
4.5.2	McIEDMA_ECX (McIConfigSet).....	98
4.5.3	McIEDMAChGroup0Priority (McIConfigSet).....	98
4.5.4	McIEDMAChGroup1Priority (McIConfigSet).....	99
4.5.5	McIEDMA_CLM (McIConfigSet).....	99
4.5.6	McIEDMA_HALT (McIConfigSet).....	99
4.5.7	McIEDMA_HOE (McIConfigSet).....	100
4.5.8	McIEDMA_ERGA (McIConfigSet).....	100
4.5.9	McIEDMA_ERCA (McIConfigSet).....	101
4.5.10	McIEDMA_EDBG (McIConfigSet).....	101
4.6	Form McIIsrAvailable.....	102
4.6.1	McIIsrName (McIIsrAvailable).....	102
4.6.2	McIIsrEnabled (McIIsrAvailable).....	102
4.7	Form DMACHannel.....	103
4.7.1	McIDMAChannelId (DMACHannel).....	103
4.7.2	DmaHwChannel (DMACHannel).....	103
4.7.3	DMACHannelPriority (DMACHannel).....	104
4.7.4	ECP (DMACHannel).....	104
4.7.5	DPA (DMACHannel).....	105
4.7.6	EMI (DMACHannel).....	105
4.7.7	McIDmaTransferCompletionNotif (DMACHannel).....	105

Section number	Title	Page
4.7.8	MclDMAChannelEnable (DMAChannel).....	106
4.7.9	MclDMAChannelTriggerEnable (DMAChannel).....	106
4.7.10	DmaSource0 (DMAChannel).....	107
4.7.11	DmaSource1 (DMAChannel).....	107
4.8	Form MclCrossbarLogicalSlavePorts.....	107
4.8.1	MclCrossbarLogicalSlavePortId (MclCrossbarLogicalSlavePorts).....	108
4.8.2	MclCrossbarHwSlavePortRef (MclCrossbarLogicalSlavePorts).....	108
4.9	Form MclCrossbarLogicalMasterPorts.....	108
4.9.1	MclCrossbarLogicalSlavePortId (MclCrossbarLogicalSlavePorts).....	109
4.9.2	MclCrossbarHwMasterPortRef (MclCrossbarLogicalMasterPorts).....	109
4.10	Form MclCrossbarInstance.....	109
4.10.1	MclCrossbarHwInstance (MclCrossbarInstance).....	110
4.10.2	Form MclCrossbarHwSlavePort.....	110
4.10.2.1	MclSlavePortNumber (MclCrossbarHwSlavePort).....	110
4.10.2.2	MclCrossbarPrioMaster0 (MclCrossbarHwSlavePort).....	111
4.10.2.3	MclCrossbarPrioMaster1 (MclCrossbarHwSlavePort).....	111
4.10.2.4	MclCrossbarPrioMaster2 (MclCrossbarHwSlavePort).....	112
4.10.2.5	MclCrossbarPrioMaster3 (MclCrossbarHwSlavePort).....	112
4.10.2.6	MclCrossbarPrioMaster4 (MclCrossbarHwSlavePort).....	112
4.10.2.7	MclCrossbarPrioMaster5 (MclCrossbarHwSlavePort).....	113
4.10.2.8	MclCrossbarPrioMaster6 (MclCrossbarHwSlavePort).....	113
4.10.2.9	MclCrossbarPrioMaster7 (MclCrossbarHwSlavePort).....	114
4.10.2.10	MclCrossbarEnableLock (MclCrossbarHwSlavePort).....	114
4.10.2.11	MclCrossbarHaltLowPrio (MclCrossbarHwSlavePort).....	114
4.10.2.12	MclCrossbarEnablePrioElevM0 (MclCrossbarHwSlavePort).....	115
4.10.2.13	MclCrossbarEnablePrioElevM1 (MclCrossbarHwSlavePort).....	115
4.10.2.14	MclCrossbarEnablePrioElevM2 (MclCrossbarHwSlavePort).....	115
4.10.2.15	MclCrossbarEnablePrioElevM3 (MclCrossbarHwSlavePort).....	116
4.10.2.16	MclCrossbarEnablePrioElevM4 (MclCrossbarHwSlavePort).....	116

Section number	Title	Page
4.10.2.17	MclCrossbarEnablePrioElevM5 (MclCrossbarHwSlavePort).....	117
4.10.2.18	MclCrossbarEnablePrioElevM6 (MclCrossbarHwSlavePort).....	117
4.10.2.19	MclCrossbarEnablePrioElevM7 (MclCrossbarHwSlavePort).....	117
4.10.2.20	MclCrossbarEnableFixedPrio (MclCrossbarHwSlavePort).....	118
4.10.2.21	MclCrossbarParkingControl (MclCrossbarHwSlavePort).....	118
4.10.2.22	MclCrossbarParkField (MclCrossbarHwSlavePort).....	119
4.10.3	Form MclCrossbarHwMasterPort.....	119
4.10.3.1	MclMasterPortNumber (MclCrossbarHwMasterPort).....	119
4.10.3.2	MclCrossbarArbitrates (MclCrossbarHwMasterPort).....	120
4.11	Form CommonPublishedInformation.....	120
4.11.1	ArReleaseMajorVersion (CommonPublishedInformation).....	121
4.11.2	ArReleaseMinorVersion (CommonPublishedInformation).....	121
4.11.3	ArReleaseRevisionVersion (CommonPublishedInformation).....	122
4.11.4	ModuleId (CommonPublishedInformation).....	122
4.11.5	SwMajorVersion (CommonPublishedInformation).....	123
4.11.6	SwMinorVersion (CommonPublishedInformation).....	123
4.11.7	SwPatchVersion (CommonPublishedInformation).....	124
4.11.8	VendorApiInfix (CommonPublishedInformation).....	124
4.11.9	VendorId (CommonPublishedInformation).....	125



# 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 User Manual describes NXP MicroController Library ( MCL ) for S32R274 .

MCL driver configuration parameters and deviations from the specification are described in MCL Driver chapter of this document. MCL driver requirements and APIs are described in the MCL driver software specification document.

### 2.1 Supported Derivatives

The software described in this document is intended 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

### Driver

#### 3.1 Requirements

MCL is a complex driver, so there are no AUTOSAR requirements regarding this module. For the S32R274 platform, the MCL module configures the DMA, DMAMUX, and AXBS functionalities.

#### 3.2 Driver Design Summary

The MCL driver configures the direct memory access, direct memory access multiplexer and the crossbar switch. Also, MCL it is a container for common functionalities: eTimer common files are included in MCL because they are used by several modules(eg. ICU, PWM). If some modules need the eTimer common files and they do not need the DMA or AXBS(crossbar) functionality, they should only include the eTimer common files, without configuring MCL.

The MCL driver has the following major features related to DMA configuration and usage:

- Source- and destination-address calculations
- Data-movement operations.
- Local memory containing transfer control descriptors for each channel.
- Configuration error checking by polling or interrupt-driven.

### 3.3 Hardware Resources

**Table 3-1. Hardware Resources for S32R274 family**

Hardware IP	Description
eDMA	Enhanced Direct Memory Access. The platform includes 1 DMA instance having 32 hardware channels.
DMAMUX	Direct Memory Access Multiplexer. The platform includes 2 DMAMUX instances, each having 16 hardware channels.
AXBS	Crossbar switch. MCL supports Data Crossbar (AXBS_0), Instruction Crossbar (AXBS_1).
eTimer	Enhanced Motor Control Timer. MCL includes the shared lower level functionalities for eTimer.

### 3.4 Deviation from requirements

None.

### 3.5 Driver Limitations

The MCL driver software have some following limitations:

- The user must not call `Mcl_Init` or `Mcl_DmaSetChannelPriority` while the transfer is active.
- MCL does not support the coherency models needed for dynamically setting Scatter gather or linking. The user shall not dynamicaly set scatter gather or linking. (dynamically set=to set while channel is in execution)
- User must not use `INT_HALF` notification if it has `BITER=1`
- `eDMA_CR` is partially implemented: THE `EMLM` bit can not be configured; the `EMLM` bit is always set to 1 during the DMA initialization.
- `eDMA_HRS` is not implemented: MCL does not provide API to retrieve this information.
- Due to errata e10452, when master ID replication is enabled, the stored ID and privilege level will change if read by another master. The user should make sure only allow the intended master ID replication core to access the `DMA_TCDn_CSR[DONE:START]` byte.

### 3.6 Driver Usage and Configuration tips

- Configure only the feature that is needed in your project: in case the DMA support is needed, configure EnableDMA as true; in case the crossbar support is needed configure MclEnableCrossbarSwitch as true; if DMA and Crossbar are not needed, the MCL module is still needed in the smcal project as MCL is a library which includes the timer shared files used by other MCAL modules(GPT,ICU,PWM) - in this case you don't need to configure MCL and you don't need to call Mcl\_Init because the timer hardware is configured and initialized by other MCAL modules.
- For the DMA feature: in case the channel half or end notification is needed the function name must be configured(MclDmaTransferCompletionNotif parameter)and the corresponding ISR must be enabled(MclIsrEnabled parameter).
- For the DMA feature: do not use INT\_HALF notification if the major iteration count is 1

### 3.7 Runtime Errors

The driver generates the following DEM errors at runtime.

**Table 3-2. Runtime Errors**

Function	Error Code	Condition triggering the error
Mcl_DmaGetGlobalErrorStatus	MCL_DMA_E_DESCRIPTOR	The DEM event MCL_DMA_E_DESCRIPTOR is reported by the software when one of the APIs Mcl_DmaGetGlobalErrorStatus or Mcl_DmaGetChannelErrorStatus is called. The APIs get the error status from hardware registers. This means it will be reported asynchronous to the occurrence of the error condition in hardware. For a synchronous error reporting, the user should configure the MCL DMA error notification. Description of the error condition in the hardware: This error is reported when the DMA TCD is incorrectly configured, this means when one of the following rules is broken: <ul style="list-style-type: none"> <li>• The addresses and offsets must be aligned on 0-modulo-transfer-size boundaries.</li> <li>• The minor loop byte count must be a multiple of the source and destination transfer sizes.</li> <li>• All source reads and destination writes must be configured to the natural boundary of the programmed transfer size respectively.</li> <li>• If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32- byte boundary.</li> <li>• If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCDn_CITER[E_LINK] bit does not equal the TCDn_BITER[E_LINK] bit. If</li> </ul>

*Table continues on the next page...*

Table 3-2. Runtime Errors (continued)

Function	Error Code	Condition triggering the error
		enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, are reported by the hardware as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported by the hardware when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported by the hardware when the link operation is serviced at minor loop completion.
Mcl_DmaGetChannelErrorStatus	MCL_DMA_E_DESCRIPTOR	Description of the error condition in the hardware: This error is reported when the DMA TCD is incorrectly configured, this means when one of the following rules is broken: <ul style="list-style-type: none"> <li>• The addresses and offsets must be aligned on 0-modulo-transfer-size boundaries.</li> <li>• The minor loop byte count must be a multiple of the source and destination transfer sizes.</li> <li>• All source reads and destination writes must be configured to the natural boundary of the programmed transfer size respectively.</li> <li>• If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32- byte boundary.</li> <li>• If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCDn_CITER[E_LINK] bit does not equal the TCDn_BITER[E_LINK] bit.</li> </ul> If enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, are reported by the hardware as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported by the hardware when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported by the hardware when the link operation is serviced at minor loop completion.
Mcl_DmaGetGlobalErrorStatus	MCL_DMA_E_ECC	The error is reported when the UCE bit of DMA_ES is set because of an uncorrectable ECC error during channel execution.
Mcl_DmaGetChannelErrorStatus	MCL_DMA_E_ECC	The error is reported when the UCE bit of DMA_ES is set because of an uncorrectable ECC error during channel execution.
Mcl_DmaGetGlobalErrorStatus	MCL_DMA_E_BUS	Description of the error condition in the hardware: <ul style="list-style-type: none"> <li>• The error condition related to this event occurs in hardware when a source bus error or a destination bus error is reported by the DMA hardware during a transfer.</li> <li>• If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and</li> </ul>

Table continues on the next page...

Table 3-2. Runtime Errors (continued)

Function	Error Code	Condition triggering the error
		current iteration count at the point of the fault. When a system bus error occurs, the channel terminates after the next transfer. Due to pipeline effect, the next transfer is already in progress when the bus error is received by the eDMA. If a bus error occurs on the last read prior to beginning the write sequence, the write executes using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel terminates due to the destination bus error.
Mcl_DmaGetChannelErrorStatus	MCL_DMA_E_BUS	Description of the error condition in the hardware: • The error condition related to this event occurs in hardware when a source bus error or a destination bus error is reported by the DMA hardware during a transfer. • If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and current iteration count at the point of the fault. When a system bus error occurs, the channel terminates after the next transfer. Due to pipeline effect, the next transfer is already in progress when the bus error is received by the eDMA. If a bus error occurs on the last read prior to beginning the write sequence, the write executes using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel terminates due to the destination bus error.
Mcl_DmaGetGlobalErrorStatus	MCL_DMA_E_PRIORITY	Description of the error condition in the hardware: A priority configuration error happens in the fixed arbitration mode and it is caused by any two channel priorities being equal within a group of channels.
Mcl_DmaGetChannelErrorStatus	MCL_DMA_E_PRIORITY	Description of the error condition in the hardware: A priority configuration error happens in the fixed arbitration mode and it is caused by any two channel priorities being equal within a group of channels.
Mcl_DmaGetGlobalErrorStatus	MCL_DMA_E_INCONSISTENCY	The DEM event MCL_DMA_E_INCONSISTENCY is reported by the software when one of the APIs Mcl_DmaGetGlobalErrorStatus or Mcl_DmaGetChannelErrorStatus is called. The APIs get the error status from hardware registers. This means it will be reported asynchronous to the occurrence of the error condition in hardware. For a synchronous error reporting, the user should configure the MCL DMA error notification. This error event is set by software if the registers

Table continues on the next page...

Table 3-2. Runtime Errors (continued)

Function	Error Code	Condition triggering the error
		<p>DMA_ES and DMA_ERR report inconsistent error information, in one of following cases:</p> <ul style="list-style-type: none"> <li>• DMA_ES reports errors for a respective channel and DMA_ERR reports no error for that channel</li> <li>• DMA_ERR reports errors for a respective channel and DMA_ES reports no error for that channel</li> <li>• DMA_ES and DMA_ERR report errors for different channels</li> </ul> <p>Error code</p> <p>MCL_DMA_E_INCONSISTENCY marks that the DMA might have met an error condition, but this is not clear because the hardware reports it in inconsistent matter.</p>
Mcl_DmaGetChannelErrorStatus	MCL_DMA_E_INCONSISTENCY	<p>The DEM event MCL_DMA_E_INCONSISTENCY is reported by the software when one of the APIs Mcl_DmaGetGlobalErrorStatus or Mcl_DmaGetChannelErrorStatus is called. The APIs get the error status from hardware registers. This means it will be reported asynchronous to the occurrence of the error condition in hardware. For a synchronous error reporting, the user should configure the MCL DMA error notification. This error event is set by software if the registers DMA_ES and DMA_ERR report inconsistent error information, in one of following cases:</p> <ul style="list-style-type: none"> <li>• DMA_ES reports errors for a respective channel and DMA_ERR reports no error for that channel</li> <li>• DMA_ERR reports errors for a respective channel and DMA_ES reports no error for that channel</li> <li>• DMA_ES and DMA_ERR report errors for different channels</li> </ul> <p>Error code</p> <p>MCL_DMA_E_INCONSISTENCY marks that the DMA might have met an error condition, but this is not clear because the hardware reports it in inconsistent matter.</p>
Mcl_DmaGetGlobalErrorStatus	MCL_DMA_E_UNRECOGNIZED	<p>The DEM event MCL_DMA_E_UNRECOGNIZED is reported by the software when one of the APIs Mcl_DmaGetGlobalErrorStatus or Mcl_DmaGetChannelErrorStatus is called. The APIs get the error status from hardware registers. This means it will be reported asynchronous to the occurrence of the error condition in hardware. For a synchronous error reporting, the user should configure the MCL DMA error notification. This error event is set by software if the registers DMA_ES and DMA_ERR report error for the same channel, but the register DMA_ES doesn't provide any error status (no ECC error, no bus error, no descriptor error, no priority error). This might happen because of issues in the hardware.</p>
Mcl_DmaGetChannelErrorStatus	MCL_DMA_E_UNRECOGNIZED	<p>The DEM event MCL_DMA_E_UNRECOGNIZED is reported by the software when one of the APIs Mcl_DmaGetGlobalErrorStatus or Mcl_DmaGetChannelErrorStatus is called. The APIs get the error status from hardware registers.</p>



**Table 3-2. Runtime Errors**

Function	Error Code	Condition triggering the error
		This means it will be reported asynchronous to the occurrence of the error condition in hardware. For a synchronous error reporting, the user should configure the MCL DMA error notification. This error event is set by software if the registers DMA_ES and DMA_ERR report error for the same channel, but the register DMA_ES doesn't provide any error status (no ECC error, no bus error, no descriptor error, no priority error). This might happen because of issues in the hardware.

## 3.8 Software specification

The following sections contains driver software specifications.

### 3.8.1 Define Reference

This chapter describes the defines supported by the MCL driver.

#### 3.8.1.1 Define MCL\_ACKNOWLEDGEINTERRUPT\_ID\_U8

API service ID for Mcl\_DmaAcknowledgeInterrupt function.

##### Details:

Parameters used when raising an error/exception

**Table 3-3. Define MCL\_ACKNOWLEDGEINTERRUPT\_ID\_U8**  
**Description**

<b>Name</b>	MCL_ACKNOWLEDGEINTERRUPT_ID_U8
<b>Initializer</b>	(uint8)0x34U

#### 3.8.1.2 Define MCL\_AR\_RELEASE\_MAJOR\_VERSION

**Implements:** Mcl\_interface

**Table 3-4. Define MCL\_AR\_RELEASE\_MAJOR\_VERSION  
Description**

<b>Name</b>	MCL_AR_RELEASE_MAJOR_VERSION
<b>Initializer</b>	4

### 3.8.1.3 Define MCL\_AR\_RELEASE\_MINOR\_VERSION

**Implements:** Mcl\_interface

**Table 3-5. Define MCL\_AR\_RELEASE\_MINOR\_VERSION  
Description**

<b>Name</b>	MCL_AR_RELEASE_MINOR_VERSION
<b>Initializer</b>	0

### 3.8.1.4 Define MCL\_AR\_RELEASE\_REVISION\_VERSION

**Implements:** Mcl\_interface

**Table 3-6. Define MCL\_AR\_RELEASE\_REVISION\_VERSION  
Description**

<b>Name</b>	MCL_AR_RELEASE_REVISION_VERSION
<b>Initializer</b>	3

### 3.8.1.5 Define MCL\_CONFIG\_CH\_ID\_U8

API service ID for Mcl\_DmaConfigChannel function.

**Details:**

Parameters used when raising an error/exception

**Table 3-7. Define MCL\_CONFIG\_CH\_ID\_U8 Description**

<b>Name</b>	MCL_CONFIG_CH_ID_U8
<b>Initializer</b>	(uint8)0x24U

### 3.8.1.6 Define MCL\_CONFIG\_LINK\_CH\_ID\_U8

API service ID for Mcl\_DmaConfigLinkedChannel function.

#### Details:

Parameters used when raising an error/exception

**Table 3-8. Define MCL\_CONFIG\_LINK\_CH\_ID\_U8 Description**

<b>Name</b>	MCL_CONFIG_LINK_CH_ID_U8
<b>Initializer</b>	(uint8)0x25U

### 3.8.1.7 Define MCL\_CONFIG\_LINK\_TCD\_ID\_U8

API service ID for Mcl\_DmaConfigLinkedTcd function.

#### Details:

Parameters used when raising an error/exception

**Table 3-9. Define MCL\_CONFIG\_LINK\_TCD\_ID\_U8 Description**

<b>Name</b>	MCL_CONFIG_LINK_TCD_ID_U8
<b>Initializer</b>	(uint8)0x28U

### 3.8.1.8 Define MCL\_CONFIG\_SCA\_CH\_ID\_U8

API service ID for Mcl\_DmaConfigScatterGatherChannel function.

#### Details:

Parameters used when raising an error/exception

**Table 3-10. Define MCL\_CONFIG\_SCA\_CH\_ID\_U8 Description**

<b>Name</b>	MCL_CONFIG_SCA_CH_ID_U8
<b>Initializer</b>	(uint8)0x27U

### 3.8.1.9 Define MCL\_CONFIG\_SCA\_LINK\_CH\_ID\_U8

API service ID for Mcl\_DmaConfigScatterGatherLinkedChannel function.

#### Details:

Parameters used when raising an error/exception

**Table 3-11. Define MCL\_CONFIG\_SCA\_LINK\_CH\_ID\_U8 Description**

<b>Name</b>	MCL_CONFIG_SCA_LINK_CH_ID_U8
<b>Initializer</b>	(uint8)0x31U

### 3.8.1.10 Define MCL\_CONFIG\_SCA\_LINK\_TCD\_ID\_U8

API service ID for Mcl\_DmaConfigScatterGatherLinkedTcd function.

#### Details:

Parameters used when raising an error/exception

**Table 3-12. Define MCL\_CONFIG\_SCA\_LINK\_TCD\_ID\_U8 Description**

<b>Name</b>	MCL_CONFIG_SCA_LINK_TCD_ID_U8
<b>Initializer</b>	(uint8)0x36U

### 3.8.1.11 Define MCL\_CONFIG\_SCA\_TCD\_ID\_U8

API service ID for Mcl\_DmaConfigScatterGatherTcd function.

**Details:**

Parameters used when raising an error/exception

**Table 3-13. Define MCL\_CONFIG\_SCA\_TCD\_ID\_U8 Description**

<b>Name</b>	MCL_CONFIG_SCA_TCD_ID_U8
<b>Initializer</b>	(uint8)0x29U

### 3.8.1.12 Define MCL\_CONFIG\_TCD\_ID\_U8

API service ID for Mcl\_DmaConfigTcd function.

**Details:**

Parameters used when raising an error/exception

**Table 3-14. Define MCL\_CONFIG\_TCD\_ID\_U8 Description**

<b>Name</b>	MCL_CONFIG_TCD_ID_U8
<b>Initializer</b>	(uint8)0x26U

### 3.8.1.13 Define MCL\_DISABLENOTIFICATION\_ID\_U8

API service ID of Mcl\_DmaDisableNotification function.

**Details:**

Parameters used when raising an error/exception

**Table 3-15. Define MCL\_DISABLENOTIFICATION\_ID\_U8 Description**

<b>Name</b>	MCL_DISABLENOTIFICATION_ID_U8
<b>Initializer</b>	(uint8)0x22U

### 3.8.1.14 Define MCL\_DMACLEARDONE\_ID\_U8

API service ID for Mcl\_DmaClearDone function.

#### Details:

Parameters used when raising an error/exception

**Table 3-16. Define MCL\_DMACLEARDONE\_ID\_U8 Description**

<b>Name</b>	MCL_DMACLEARDONE_ID_U8
<b>Initializer</b>	(uint8)0x30U

### 3.8.1.15 Define MCL\_DMAGETCHANNELTCDADDRESS\_ID\_U8

API service ID for Mcl\_DmaGetChannelTcdAddress function.

#### Details:

Parameters used when raising an error/exception

**Table 3-17. Define MCL\_DMAGETCHANNELTCDADDRESS\_ID\_U8 Description**

<b>Name</b>	MCL_DMAGETCHANNELTCDADDRESS_ID_U8
<b>Initializer</b>	(uint8)0x2FU

### 3.8.1.16 Define MCL\_DMAGETINTERRUPTREQUEST\_ID\_U8

API service ID for Mcl\_DmaGetInterruptRequest function.

#### Details:

Parameters used when raising an error/exception

**Table 3-18. Define MCL\_DMAGETINTERRUPTREQUEST\_ID\_U8 Description**

<b>Name</b>	MCL_DMAGETINTERRUPTREQUEST_ID_U8
<b>Initializer</b>	(uint8)0x33U

### 3.8.1.17 Define MCL\_DMAGETPHYSICALCHANNEL\_ID\_U8

API service ID for Mcl\_DmaGetPhysicalChannel function.

#### Details:

Parameters used when raising an error/exception

**Table 3-19. Define MCL\_DMAGETPHYSICALCHANNEL\_ID\_U8 Description**

<b>Name</b>	MCL_DMAGETPHYSICALCHANNEL_ID_U8
<b>Initializer</b>	(uint8)0x35U

### 3.8.1.18 Define MCL\_DMATCDGETINTMAJ\_ID\_U8

API service ID for Mcl\_DmaTcdGetIntMaj function.

#### Details:

Parameters used when raising an error/exception

**Table 3-20. Define MCL\_DMATCDGETINTMAJ\_ID\_U8 Description**

<b>Name</b>	MCL_DMATCDGETINTMAJ_ID_U8
<b>Initializer</b>	(uint8)0x32U

### 3.8.1.19 Define MCL\_E\_ALREADY\_INITIALIZED\_U8

API Mcl\_Dma\_Init service called when the Mcl driver and the Hardware are already initialized.

**Implements:** Mcl\_ErrorCodes\_define

**Table 3-21. Define MCL\_E\_ALREADY\_INITIALIZED\_U8**  
Description

<b>Name</b>	MCL_E_ALREADY_INITIALIZED_U8
<b>Initializer</b>	(uint8)0x0D

### 3.8.1.20 Define MCL\_E\_INVALID\_CHANNEL\_U8

API service used with a channel out of range.

**Implements:** Mcl\_ErrorCodes\_define

**Table 3-22. Define MCL\_E\_INVALID\_CHANNEL\_U8 Description**

<b>Name</b>	MCL_E_INVALID_CHANNEL_U8
<b>Initializer</b>	(uint8)0x0B

### 3.8.1.21 Define MCL\_E\_PARAM\_CONFIG\_U8

API Mcl\_Init service called with wrong parameter.

**Implements:** Mcl\_ErrorCodes\_define

**Table 3-23. Define MCL\_E\_PARAM\_CONFIG\_U8 Description**

<b>Name</b>	MCL_E_PARAM_CONFIG_U8
<b>Initializer</b>	(uint8)0x12U

### 3.8.1.22 Define MCL\_E\_PARAM\_NOTIFICATION\_NULL\_U8

NULL function is configured as notification callback.

**Details:**

Will be generated when a NULL function is configured as notification callback for one DMA channel and Mcl\_Dma\_EnableNotification is called for that channel



**Implements:** Mcl\_ErrorCodes\_define

**Table 3-24. Define MCL\_E\_PARAM\_NOTIFICATION\_NULL\_U8 Description**

<b>Name</b>	MCL_E_PARAM_NOTIFICATION_NULL_U8
<b>Initializer</b>	(uint8)0x10U

### 3.8.1.23 Define MCL\_E\_PARAM\_POINTER\_U8

All API's having pointers as parameters shall return this error if called with with a NULL value.

**Implements:** Mcl\_ErrorCodes\_define

**Table 3-25. Define MCL\_E\_PARAM\_POINTER\_U8 Description**

<b>Name</b>	MCL_E_PARAM_POINTER_U8
<b>Initializer</b>	(uint8)0x0A

### 3.8.1.24 Define MCL\_E\_PARAM\_VINFO\_U8

API Mcl\_GetVersionInfo is called and the parameter versioninfo is is invalid ( e.g. NULL )

**Implements:** Mcl\_ErrorCodes\_define

**Table 3-26. Define MCL\_E\_PARAM\_VINFO\_U8 Description**

<b>Name</b>	MCL_E_PARAM_VINFO_U8
<b>Initializer</b>	(uint8)0x0F

### 3.8.1.25 Define MCL\_E\_UNEXPECTED\_ISR\_U8

Generated when an ISR has been triggered 1. when the driver is not initialized 2. for a Hw channel that is not used by any logic channel 3. for a logic channel that has no notification configured.

**Details:**

Errors and exceptions that will be detected by the MCL driver

**Implements:** Mcl\_ErrorCodes\_define

**Table 3-27. Define MCL\_E\_UNEXPECTED\_ISR\_U8 Description**

<b>Name</b>	MCL_E_UNEXPECTED_ISR_U8
<b>Initializer</b>	(uint8)0x11U

### 3.8.1.26 Define MCL\_E\_UNINIT\_U8

API service used without module initialization.

**Implements:** Mcl\_ErrorCodes\_define

**Table 3-28. Define MCL\_E\_UNINIT\_U8 Description**

<b>Name</b>	MCL_E_UNINIT_U8
<b>Initializer</b>	(uint8)0x0C

### 3.8.1.27 Define MCL\_ENABLENOTIFICATION\_ID\_U8

API service ID of Mcl\_DmaEnableNotification function.

**Details:**

Parameters used when raising an error/exception

**Table 3-29. Define MCL\_ENABLENOTIFICATION\_ID\_U8 Description**

<b>Name</b>	MCL_ENABLENOTIFICATION_ID_U8
<b>Initializer</b>	(uint8)0x21U

### 3.8.1.28 Define MCL\_GETVERSIONINFO\_ID\_U8

API service ID for Mcl\_GetVersionInfo function.

#### Details:

Parameters used when raising an error/exception

**Table 3-30. Define MCL\_GETVERSIONINFO\_ID\_U8 Description**

<b>Name</b>	MCL_GETVERSIONINFO_ID_U8
<b>Initializer</b>	(uint8)0x20U

### 3.8.1.29 Define MCL\_GET\_GLOBAL\_ERR\_STATUS\_ID\_U8

API service ID for Mcl\_DmaGetGlobalErrorStatus function.

#### Details:

Parameters used when raising an error/exception

**Table 3-31. Define MCL\_GET\_GLOBAL\_ERR\_STATUS\_ID\_U8 Description**

<b>Name</b>	MCL_GET_GLOBAL_ERR_STATUS_ID_U8
<b>Initializer</b>	(uint8)0x52U

### 3.8.1.30 Define MCL\_GET\_CH\_ERR\_STATUS\_ID\_U8

API service ID for Mcl\_DmaGetChannelErrorStatus function.

#### Details:

Parameters used when raising an error/exception

**Table 3-32. Define MCL\_GET\_CH\_ERR\_STATUS\_ID\_U8 Description**

<b>Name</b>	MCL_GET_CH_ERR_STATUS_ID_U8
<b>Initializer</b>	(uint8)0x53U

### 3.8.1.31 Define MCL\_DMA\_NO\_CHANNEL\_U16

For getting the DMA error status, the define is used when no channel should be reported.

**Table 3-33. Define MCL\_DMA\_NO\_CHANNEL\_U16 Description**

<b>Name</b>	MCL_DMA_NO_CHANNEL_U16
<b>Initializer</b>	65535U

### 3.8.1.32 Define MCL\_DMA\_CHANNEL\_NOT\_CONFIGURED\_U8

For getting the DMA error status, the define is used when no channel should be reported.

**Table 3-34. Define MCL\_DMA\_CHANNEL\_NOT\_CONFIGURED\_U8 Description**

<b>Name</b>	MCL_DMA_CHANNEL_NOT_CONFIGURED_U8
<b>Initializer</b>	255U

### 3.8.1.33 Define MCL\_INIT\_ID\_U8

API service ID for Mcl\_Init function.

#### **Details:**

Parameters used when raising an error/exception

**Table 3-35. Define MCL\_INIT\_ID\_U8 Description**

<b>Name</b>	MCL_INIT_ID_U8
<b>Initializer</b>	(uint8)0x23U

### 3.8.1.34 Define MCL\_MODULE\_ID

**Implements:** Mcl\_interface

**Table 3-36. Define MCL\_MODULE\_ID Description**

<b>Name</b>	MCL_MODULE_ID
<b>Initializer</b>	255

### 3.8.1.35 Define MCL\_SET\_PRI\_ID\_U8

API service ID for Mcl\_DmaSetChannelPriority function.

#### Details:

Parameters used when raising an error/exception

**Table 3-37. Define MCL\_SET\_PRI\_ID\_U8 Description**

<b>Name</b>	MCL_SET_PRI_ID_U8
<b>Initializer</b>	(uint8)0x2AU

### 3.8.1.36 Define MCL\_START\_CH\_ID\_U8

API service ID for Mcl\_DmaStartChannel function.

#### Details:

Parameters used when raising an error/exception

**Table 3-38. Define MCL\_START\_CH\_ID\_U8 Description**

<b>Name</b>	MCL_START_CH_ID_U8
<b>Initializer</b>	(uint8)0x2BU

### 3.8.1.37 Define MCL\_SW\_MAJOR\_VERSION

Implements: Mcl\_interface

**Table 3-39. Define MCL\_SW\_MAJOR\_VERSION**  
Description

<b>Name</b>	MCL_SW_MAJOR_VERSION
<b>Initializer</b>	1

### 3.8.1.38 Define MCL\_SW\_MINOR\_VERSION

**Implements:** Mcl\_interface

**Table 3-40. Define MCL\_SW\_MINOR\_VERSION**  
Description

<b>Name</b>	MCL_SW_MINOR_VERSION
<b>Initializer</b>	0

### 3.8.1.39 Define MCL\_SW\_PATCH\_VERSION

**Implements:** Mcl\_interface

**Table 3-41. Define MCL\_SW\_PATCH\_VERSION**  
Description

<b>Name</b>	MCL_SW_PATCH_VERSION
<b>Initializer</b>	0

### 3.8.1.40 Define MCL\_TRANSF\_ACTIVE\_ID\_U8

API service ID for Mcl\_DmaIsTransferActive function.

**Details:**

Parameters used when raising an error/exception

**Table 3-42. Define MCL\_TRANSF\_ACTIVE\_ID\_U8 Description**

<b>Name</b>	MCL_TRANSF_ACTIVE_ID_U8
<b>Initializer</b>	(uint8)0x2DU

### 3.8.1.41 Define MCL\_TRANSF\_COMPL\_ID\_U8

API service ID for Mcl\_DmaIsTransferCompleted function.

#### Details:

Parameters used when raising an error/exception

**Table 3-43. Define MCL\_TRANSF\_COMPL\_ID\_U8 Description**

<b>Name</b>	MCL_TRANSF_COMPL_ID_U8
<b>Initializer</b>	(uint8)0x2CU

### 3.8.1.42 Define MCL\_VENDOR\_ID

Implements: Mcl\_interface

**Table 3-44. Define MCL\_VENDOR\_ID Description**

<b>Name</b>	MCL_VENDOR_ID
<b>Initializer</b>	43

### 3.8.1.43 Define DMA\_TCD\_DONE\_U8

TCD Word8 bit masks for flags.

**Table 3-45. Define DMA\_TCD\_DONE\_U8 Description**

<b>Name</b>	DMA_TCD_DONE_U8
<b>Initializer</b>	((uint8)0x80U)

### 3.8.1.44 Define DMA\_TCD\_ACTIVE\_U8

TCD Word8 bit masks for flags.

**Table 3-46. Define DMA\_TCD\_ACTIVE\_U8 Description**

<b>Name</b>	DMA_TCD_ACTIVE_U8
<b>Initializer</b>	((uint8)0x40U)

### 3.8.1.45 Define DMA\_TCD\_MAJOR\_E\_LINK\_U8

TCD Word8 bit masks for flags.

**Table 3-47. Define DMA\_TCD\_MAJOR\_E\_LINK\_U8 Description**

<b>Name</b>	DMA_TCD_MAJOR_E_LINK_U8
<b>Initializer</b>	((uint8)0x20U)

### 3.8.1.46 Define DMA\_TCD\_E\_SG\_U8

TCD Word8 bit masks for flags.

**Table 3-48. Define DMA\_TCD\_E\_SG\_U8 Description**

<b>Name</b>	DMA_TCD_E_SG_U8
<b>Initializer</b>	((uint8)0x10U)

### 3.8.1.47 Define DMA\_TCD\_DISABLE\_REQ\_U8

TCD Word8 bit masks for flags.

**Table 3-49. Define DMA\_TCD\_DISABLE\_REQ\_U8 Description**

<b>Name</b>	DMA_TCD_DISABLE_REQ_U8
<b>Initializer</b>	((uint8)0x08U)

### 3.8.1.48 Define DMA\_TCD\_INT\_HALF\_U8

TCD Word8 bit masks for flags.



**Table 3-50. Define DMA\_TCD\_INT\_HALF\_U8 Description**

<b>Name</b>	DMA_TCD_INT_HALF_U8
<b>Initializer</b>	((uint8)0x04U)

### 3.8.1.49 Define DMA\_TCD\_INT\_MAJOR\_U8

TCD Word8 bit masks for flags.

**Table 3-51. Define DMA\_TCD\_INT\_MAJOR\_U8 Description**

<b>Name</b>	DMA_TCD_INT_MAJOR_U8
<b>Initializer</b>	((uint8)0x02U)

### 3.8.1.50 Define DMA\_TCD\_START\_U8

TCD Word8 bit masks for flags.

**Table 3-52. Define DMA\_TCD\_START\_U8 Description**

<b>Name</b>	DMA_TCD_START_U8
<b>Initializer</b>	((uint8)0x01U)

### 3.8.1.51 Define MCL\_GET\_CRT\_ITER\_CH\_ID\_U8

**Implements:** Mcl\_interface

**Table 3-53. Define MCL\_GET\_CRT\_ITER\_CH\_ID\_U8 Description**

<b>Name</b>	MCL_GET_CRT_ITER_CH_ID_U8
<b>Initializer</b>	(uint8)0x4E

### 3.8.1.52 Define MCL\_GET\_STRT\_ITER\_CH\_ID\_U8

**Implements:** Mcl\_interface

**Table 3-54. Define MCL\_GET\_STRT\_ITER\_CH\_ID\_U8 Description**

<b>Name</b>	MCL_GET_STRT_ITER_CH_ID_U8
<b>Initializer</b>	(uint8)0x50

### 3.8.1.53 Define MCL\_UPDATE\_DEST\_CH\_ID\_U8

**Implements:** Mcl\_interface

**Table 3-55. Define MCL\_UPDATE\_DEST\_CH\_ID\_U8 Description**

<b>Name</b>	MCL_UPDATE_DEST_CH_ID_U8
<b>Initializer</b>	(uint8)0x4F

### 3.8.1.54 Define MCL\_UPDATE\_ITER\_ID\_U8

**Implements:** Mcl\_interface

**Table 3-56. Define MCL\_UPDATE\_ITER\_ID\_U8 Description**

<b>Name</b>	MCL_UPDATE_ITER_ID_U8
<b>Initializer</b>	(uint8)0x4D

## 3.8.2 Enum Reference

This chapter describes the enums supported by the MCL driver.

### 3.8.2.1 Enumeration Mcl\_DmaTransferNotifType

Dma notification configuration structure.

**Implements:** Mcl\_DmaTransferNotifType\_enum

**Table 3-57. Enumeration Mcl\_DmaTransferNotifType Values**

Name	Initializer	Description
MCL_DMA_TRANSFER_COMPLETE	0	A notification will be generated when major iteration count completes.
MCL_DMA_TRANSFER_HALF_COMPLETE		A notification will be generated when major counter is half complete.

### 3.8.2.2 Enumeration Mcl\_DmaSizeType

Dma transfer size structure.

**Implements:** Mcl\_DmaTransferNotifType\_enum

**Table 3-58. Enumeration Mcl\_DmaSizeType Values**

Name	Initializer	Description
DMA_SIZE_1BYTE	0	Transfer size 1 byte.
DMA_SIZE_2BYTES	1	Transfer size 2 bytes.
DMA_SIZE_4BYTES	2	Transfer size 4 bytes.
DMA_SIZE_8BYTES	3	Transfer size 16 bytes.
DMA_SIZE_32BYTES	5	Transfer size 32 bytes.

### 3.8.2.3 Enumeration Mcl\_DmaRequestType

Mcl\_DmaRequestType provides the request for APIs which get info from hardware.

**Implements:** Mcl\_DmaRequestType\_enum

**Table 3-59. Enumeration Mcl\_DmaRequestType Values**

Name	Initializer	Description
MCL_DMA_GET_ERR	0	Indicates if an error request.
MCL_DMA_GET_INT	1	Indicates if an interrupt request.

### 3.8.2.4 Enumeration Mcl\_DmaChannelErrorStatusType

Mcl\_DmaChannelErrorStatusType provides the numeric ID of a Mcl DMA error.

**Implements:** Mcl\_DmaChannelErrorStatusType\_enum

**Table 3-60. Enumeration Mcl\_DmaChannelErrorStatusType Values**

Name	Initializer	Description
MCL_DMA_NO_ERROR	0	Mcl DMA with no error.
MCL_DMA_HW_INCONSISTENCY_ERROR	1	Mcl DMA with hardware inconsistency error.
MCL_DMA_ECC_ERROR	2	Mcl DMA with ecc error.
MCL_DMA_BUS_ERROR	3	Mcl DMA with bus error.
MCL_DMA_DESCRIPTOR_ERROR	4	Mcl DMA with descriptor error.
MCL_DMA_PRIORITY_ERROR	5	Mcl DMA with priority error.
MCL_DMA_UNRECOGNIZED_ERROR	6	Mcl DMA with unrecognized error.
MCL_DMA_MEM_SYNC_ERROR	7	Mcl DMA CACHE synchronization error, timeout occurred and CACHE command was not performed. This error code is reported via the error notification only.

## 3.8.3 Function Reference

This chapter describes the functions supported by the MCL driver.

### 3.8.3.1 Function Mcl\_DmaAcknowledgeInterrupt

This function acknowledges the interrupt for the channel passed as parameter.

#### Details:

The function Mcl\_DmaAcknowledgeInterrupt shall acknowledge the interrupt for the channel passed as parameter. If development error detection for the Mcl module is enabled:

- The Mcl functions shall check the parameter ChannelNumber and raise development error MCL\_E\_PARAM\_CHANNEL if the parameter ChannelNumber is invalid.

If development error detection for the Mcl module is enabled, when a development error occurs, the corresponding Mcl function shall:

- Report the error to the Development Error Tracer.
- Skip the desired functionality in order to avoid any corruptions of data or hardware registers (this means leave the function without any actions).

If the MclDevErrorDetect switch is enabled, API parameter checking is enabled. The detailed description of the detected errors can be found in chapter

If development error detection for the Mcl module is enabled, if any function (except Mcl\_Init) is called before Mcl\_Init has been called, the called function shall raise development error MCL\_E\_UNINIT.

**Return:** void .

**Implements:** Mcl\_DmaAcknowledgeInterrupt\_Activity

**Prototype:** void Mcl\_DmaAcknowledgeInterrupt(Mcl\_ChannelType ChannelNumber);

**Table 3-61. Mcl\_DmaAcknowledgeInterrupt Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	ChannelNumber	input	- Channel id .

### 3.8.3.2 Function Mcl\_DmaConfigChannel

This function configures a DMA Channel.

**Details:**

This function is reentrant and configures the specified DMA channel

**Return:** void.

**Pre:** Mcl\_Dma\_Init must be called before.

**Implements:** Mcl\_DmaConfigChannel\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaConfigChannel(Mcl\_ChannelType dma\_channel, const Mcl\_DmaTcdAttributesType \*config\_descriptor);

**Table 3-62. Mcl\_DmaConfigChannel Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	dma_channel	input	Numeric identifier of the DMA channel.
constMcl_DmaTcdAttributesType*	config_descriptor	input	Pointer to the channel's descriptor attributes.

### 3.8.3.3 Function Mcl\_DmaConfigLinkedChannel

This function configures linked DMA Channel.

#### Details:

This function is reentrant and configures the specified linked DMA channel

**Return:** void.

**Pre:** Mcl\_Dma\_Init must be called before.

**Implements:** Mcl\_DmaConfigLinkedChannel\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaConfigLinkedChannel(Mcl\_ChannelType dma\_channel, const Mcl\_DmaTcdAttributesType \*config\_descriptor, Mcl\_ChannelType next\_channel);

**Table 3-63. Mcl\_DmaConfigLinkedChannel Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	dma_channel	input	Numeric identifier of the DMA channel.
constMcl_DmaTcdAttributesType*	config_descriptor	input	Pointer to the channel's descriptor attributes.
Mcl_ChannelType	next_channel	input	Numeric identifier of the next DMA channel.

### 3.8.3.4 Function Mcl\_DmaConfigLinkedTcd

This function configures a linked DMA Tcd.

#### Details:

This function is reentrant and configures a linked DMA Tcd

**Return:** void.

**Pre:** Mcl\_Dma\_Init must be called before.

**Implements:** Mcl\_DmaConfigLinkedTcd\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaConfigLinkedTcd(Mcl\_DmaTcdType \*pTcdAddress, const Mcl\_DmaTcdAttributesType \*config\_descriptor, Mcl\_ChannelType next\_channel);

**Table 3-64. Mcl\_DmaConfigLinkedTcd Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	Pointer to the TCD to be configured.
constMcl_DmaTcdAttributesType*	config_descriptor	input	Pointer to the channel's descriptor attributes.
Mcl_ChannelType	next_channel	input	Numeric identifier of the next DMA channel.

### 3.8.3.5 Function Mcl\_DmaConfigScatterGatherChannel

This function configures a scatter gather DMA channel.

**Details:**

This function is reentrant and configures the specified scatter gather DMA Channel

**Return:** void.

**Pre:** Mcl\_Dma\_Init must be called before.

**Implements:** Mcl\_DmaConfigScatterGatherChannel\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaConfigScatterGatherChannel(Mcl\_ChannelType dma\_channel, const Mcl\_DmaTcdAttributesType \*config\_descriptor, Mcl\_DmaTcdType \*pNext\_tcd);

**Table 3-65. Mcl\_DmaConfigScatterGatherChannel Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	dma_channel	input	Numeric identifier of the DMA channel.
constMcl_DmaTcdAttributesType*	config_descriptor	input	Pointer to the channel's descriptor attributes.
Mcl_DmaTcdType*	pNext_tcd	input	Pointer to the next TCD.

### 3.8.3.6 Function Mcl\_DmaConfigScatterGatherLinkedChannel

This function configures a scatter gather DMA channel with linking.

**Details:**

This function is reentrant and configures the specified scatter gather DMA Channel and linking.

**Return:** void.

**Pre:** Mcl\_Dma\_Init must be called before.

**Implements:** Mcl\_DmaConfigScatterGatherLinkedChannel\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaConfigScatterGatherLinkedChannel(Mcl\_ChannelType dma\_channel, const Mcl\_DmaTcdAttributesType \*config\_descriptor, Mcl\_DmaTcdType \*pNext\_tcd, Mcl\_ChannelType next\_channel);

**Table 3-66. Mcl\_DmaConfigScatterGatherLinkedChannel Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	dma_channel	input	Numeric identifier of the DMA channel.
constMcl_DmaTcdAttributesType*	config_descriptor	input	Pointer to the channel's descriptor attributes.
Mcl_DmaTcdType*	pNext_tcd	input	Pointer to the TCD address used for scatter gather.
Mcl_ChannelType	next_channel	input	Channel used for link.

### 3.8.3.7 Function Mcl\_DmaConfigScatterGatherLinkedTcd

This function configures a scatter gather DMA TCD with linking.

**Details:**

This function is reentrant and configures the specified scatter gather DMA Channel and linking.

**Return:** void.



**Pre:** Mcl\_Dma\_Init must be called before.

**Implements:** Mcl\_DmaConfigScatterGatherLinkedTcd\_Activity

**Violates:** Identifier clash.

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaConfigScatterGatherLinkedTcd(Mcl\_DmaTcdType \*pTcdAddress, const Mcl\_DmaTcdAttributesType \*config\_descriptor, Mcl\_DmaTcdType \*pNext\_tcd, Mcl\_ChannelType next\_channel);

**Table 3-67. Mcl\_DmaConfigScatterGatherLinkedTcd Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	Tcd address used for configuring SGA with linking.
constMcl_DmaTcdAttributesType*	config_descriptor	input	Pointer to the channel's descriptor attributes.
Mcl_DmaTcdType*	pNext_tcd	input	Pointer to the TCD address used for scatter gather.
Mcl_ChannelType	next_channel	input	Channel used for link.

### 3.8.3.8 Function Mcl\_DmaConfigScatterGatherTcd

This function configures a linked scatter gather DMA Tcd.

**Details:**

This function is reentrant and configures a linked scatter gather DMA Tcd

**Return:** void.

**Pre:** Mcl\_Dma\_Init must be called before.

**Implements:** Mcl\_DmaConfigScatterGatherTcd\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaConfigScatterGatherTcd(Mcl\_DmaTcdType \*pTcdAddress, const Mcl\_DmaTcdAttributesType \*config\_descriptor, Mcl\_DmaTcdType \*pNext\_tcd);

**Table 3-68. Mcl\_DmaConfigScatterGatherTcd Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	Pointer to the TCD to be configured.

*Table continues on the next page...*

**Table 3-68. Mcl\_DmaConfigScatterGatherTcd Arguments (continued)**

Type	Name	Direction	Description
constMcl_DmaTcdAttributesType*	config_descriptor	input	Pointer to the channel's descriptor attributes.
Mcl_DmaTcdType*	pNext_tcd	input	Pointer to the next TCD.

### 3.8.3.9 Function Mcl\_DmaConfigTcd

This function configures a DMA Tcd.

**Details:**

This function is reentrant and configures the specified DMA Tcd

**Return:** void.

**Pre:** Mcl\_Dma\_Init must be called before.

**Implements:** Mcl\_DmaConfigTcd\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaConfigTcd(Mcl\_DmaTcdType \*pTcdAddress, const Mcl\_DmaTcdAttributesType \*config\_descriptor);

**Table 3-69. Mcl\_DmaConfigTcd Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	Pointer to the TCD to be configured.
constMcl_DmaTcdAttributesType*	config_descriptor	input	Pointer to the channel's descriptor attributes.

### 3.8.3.10 Function Mcl\_DmaEnableNotification

This function enables the user notifications at transfer completion.

**Details:**

The function Mcl\_Dma\_EnableNotification shall enable the DMA completion notification or half-completion of a transfer according to notification parameter.

**Return:** void .

**Pre:** Mcl\_Dma\_Init must be called before.

**Implements:** Mcl\_DmaEnableNotification\_Activity

**Prototype:** void Mcl\_DmaEnableNotification(Mcl\_ChannelType ChannelNumber,  
Mcl\_DmaTransferNotifType Notification);

**Table 3-70. Mcl\_DmaConfigChannel Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	dma_channel	input	Numeric identifier of the DMA channel .
Mcl_DmaTransferNotifType	Notification	input	Notification type to be enabled

### 3.8.3.11 Function Mcl\_DmaDisableNotification

This function disables the user notifications at transfer completion.

**Details:**

The function Mcl\_Dma\_DisableNotification shall disable the DMA completion notification or half-completion of a transfer.

**Return:** void .

**Pre:** Mcl\_Dma\_Init must be called before.

**Implements:** Mcl\_DmaDisableNotification\_Activity

**Prototype:** void Mcl\_DmaDisableNotification(Mcl\_ChannelType ChannelNumber);

**Table 3-71. Mcl\_DmaConfigChannel Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	dma_channel	input	Numeric identifier of the DMA channel .

### 3.8.3.12 Function Mcl\_DmaEnableHwRequest

Mcl\_DmaEnableHwRequest.

**Details:**

This function is used for enabling the hardware request for a given Mcl channel.

**Return:** Mcl\_DmaChannelType.

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaEnableHwRequest (Mcl\_ChannelType ChannelNumber);

**Table 3-72. Mcl\_DmaEnableHwRequest Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	ChannelNumber	input	- Mcl Channel for hardware request enabling.

### 3.8.3.13 Function Mcl\_DmaDisableHwRequest

Mcl\_DmaDisableHwRequest.

**Details:**

This function is used for disabling the hardware request for a given Mcl channel.

**Return:** Mcl\_DmaChannelType .

**Implements:** Mcl\_DmaDisableHwRequest\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaDisableHwRequest (Mcl\_ChannelType ChannelNumber);

**Table 3-73. Mcl\_DmaDisableHwRequest Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	ChannelNumber	input	- Mcl Channel for hardware request disabling.

### 3.8.3.14 Function Mcl\_DmaGetChannelTcdAddress

Mcl\_DmaGetChannelTcdAddress.

**Details:**

This function is used for getting the translation between a Mcl channel and the address for the corresponding tcd.

**Return:** The address of the TCD for the channel given as parameter.

**Implements:** Mcl\_DmaGetChannelTcdAddress\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** `Mcl_DmaTcdType Mcl_DmaGetChannelTcdAddress(Mcl_ChannelType ChannelNumber);`

**Table 3-74. Mcl\_DmaGetChannelTcdAddress Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	ChannelNumber	input	- Mcl Channel for which notification should be called.

### 3.8.3.15 Function Mcl\_DmaGetInterruptRequest

Mcl\_DmaGetInterruptRequest.

**Details:**

This function is used for getting the interrupt request for the specified channel

**Return:** boolean.

**Implements:** Mcl\_DmaGetInterruptRequest\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** `boolean Mcl_DmaGetInterruptRequest(Mcl_ChannelType ChannelNumber);`

**Table 3-75. Mcl\_DmaGetInterruptRequest Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	ChannelNumber	input	- Mcl Channel for getting interrupt state.

### 3.8.3.16 Function Mcl\_DmaGetPhysicalChannel

Mcl\_DmaGetPhysicalChannel.

**Details:**

This function is used for getting the physical DMA channel for a given Mcl channel.

**Return:** .

**Implements:** Mcl\_DmaGetPhysicalChannel\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** `Mcl_DmaChannelType Mcl_DmaGetPhysicalChannel(Mcl_ChannelType ChannelNumber);`

**Table 3-76. Mcl\_DmaGetPhysicalChannel Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	ChannelNumber	input	- Mcl Channel for getting the physical DMA channel.

### 3.8.3.17 Function Mcl\_DmIsTransferActive

This function checks if a DMA transfer is active.

**Details:**

This function is reentrant and checks if a DMA transfer is active

**Return:** boolean.

**Pre:** .

**Implements:** Mcl\_DmaIsTransferActive\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** `boolean Mcl_DmaIsTransferActive(Mcl_ChannelType nChannel);`

**Table 3-77. Mcl\_DmIsTransferActive Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	nChannel	input	Numeric identifier of the DMA channel.

### 3.8.3.18 Function Mcl\_DmIsTransferCompleted

This function checks if the DMA transfer is completed.

**Details:**

This function is reentrant and checks if the DMA transfer is completed

**Return:** boolean.

**Pre:** .

**Implements:** Mcl\_DmaIsTransferCompleted\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** `boolean Mcl_DmaIsTransferCompleted(Mcl_ChannelType nChannel);`

**Table 3-78. Mcl\_DmaIsTransferCompleted Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	nChannel	input	Numeric identifier of the DMA channel.

### 3.8.3.19 Function Mcl\_DmaSetChannelPriority

This function sets the priority for the specified DMA Channel.

**Details:**

This function is reentrant and sets the priority for the specified DMA Channel

**Return:** void.

**Pre:** .

**Implements:** Mcl\_DmaSetChannelPriority\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** `void Mcl_DmaSetChannelPriority(Mcl_ChannelType nChannel, Mcl_DmaPriorityType nPriority);`

**Table 3-79. Mcl\_DmaSetChannelPriority Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	nChannel	input	Numeric identifier of the DMA channel.
Mcl_DmaPriorityType	nPriority	input	Value for the priority.

### 3.8.3.20 Function Mcl\_DmaStartChannel

This function starts the specified DMA Channel.

**Details:**

This function is reentrant and starts the specified DMA Channel

**Return:** void.

**Pre:** .

**Implements:** Mcl\_DmaStartChannel\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaStartChannel(Mcl\_ChannelType nChannel);

**Table 3-80. Mcl\_DmaStartChannel Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	nChannel	input	Numeric identifier of the DMA channel.

### 3.8.3.21 Function Mcl\_DmaTcdClearDone

Mcl\_DmaTcdClearDone.

**Details:**

This function is used for setting the Channel Done, Channel Active, Enable channel-to-channel linking on major loop complete, Enable Scatter Gather Processing, Disable Request, Enable an interrupt when major counter is half complete, Enable an interrupt when major iteration count completes, Channel Start flags for a TCD based on the address of the TCD.

**Return:** .

**Implements:** Mcl\_DmaTcdClearDone\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaTcdClearDone(Mcl\_ChannelType nChannel);



**Table 3-81. Mcl\_DmaTcdClearDone Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	nChannel	input	- Channel number for clearing DONE bit.

### 3.8.3.22 Function Mcl\_DmaTcdClearIntMaj

Mcl\_DmaTcdClearIntMaj.

#### Details:

This function disables the interrupts when major iteration count completes for a TCD based on the address of the TCD.

**Return:** .

**Implements:** Mcl\_DmaTcdClearIntMaj\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaTcdClearIntMaj(Mcl\_DmaTcdType \*pTcdAddress);

**Table 3-82. Mcl\_DmaTcdClearIntMaj Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	- Address for the TCD.

### 3.8.3.23 Function Mcl\_DmaTcdGetDaddr

Mcl\_DmaTcdGetDaddr.

#### Details:

This function is used for getting the DADDR for a TCD based on the address of the TCD.

**Return:** Value for DADDR.

**Implements:** Mcl\_DmaTcdGetDaddr\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** uint32 Mcl\_DmaTcdGetDaddr(Mcl\_DmaTcdType \*pTcdAddress);

**Table 3-83. Mcl\_DmaTcdGetDaddr Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	- Address for the TCD.

### 3.8.3.24 Function Mcl\_DmaTcdGetFlags

Mcl\_DmaTcdGetFlags.

#### **Details:**

This function is used for getting the Channel Done, Channel Active, Enable channel-to-channel linking on major loop complete, Enable Scatter Gather Processing, Disable Request, Enable an interrupt when major counter is half complete, Enable an interrupt when major iteration count completes, Channel Start flags for a TCD based on the address of the TCD.

**Return:** Value for all the flags.

**Implements:** Mcl\_DmaTcdGetFlags\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** `uint8 Mcl_DmaTcdGetFlags(Mcl_DmaTcdType *pTcdAddress);`

**Table 3-84. Mcl\_DmaTcdGetFlags Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	- Address for the TCD.

### 3.8.3.25 Function Mcl\_DmaTcdGetIntMaj

Mcl\_DmaTcdGetIntMaj.

#### **Details:**

This function returns TRUE if the interrupts were enabled and FALSE if interrupts were disabled for the corresponding channel.

**Return:** boolean.

**Implements:** Mcl\_DmaTcdGetIntMaj\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** `boolean Mcl_DmaTcdGetIntMaj (Mcl_ChannelType ChannelNumber);`

**Table 3-85. Mcl\_DmaTcdGetIntMaj Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	ChannelNumber	input	- Mcl Channel for getting interrupt state.

### 3.8.3.26 Function Mcl\_DmaTcdGetIterCount

Mcl\_DmaTcdGetIterCount.

**Details:**

This function is used for getting the CITER for a TCD based on the address of the TCD.

**Return:** Value for CITER.

**Implements:** Mcl\_DmaTcdGetIterCount\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** `uint16 Mcl_DmaTcdGetIterCount (Mcl_DmaTcdType *pTcdAddress);`

**Table 3-86. Mcl\_DmaTcdGetIterCount Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	- Address for the TCD.

### 3.8.3.27 Function Mcl\_DmaTcdGetSaddr

Mcl\_DmaTcdGetSaddr.

**Details:**

This function is used for getting the SADDR for a TCD based on the address of the TCD.

**Return:** Value for SADDR.

**Implements:** Mcl\_DmaTcdGetSaddr\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** `uint32 Mcl_DmaTcdGetSaddr(Mcl_DmaTcdType *pTcdAddress);`

**Table 3-87. Mcl\_DmaTcdGetSaddr Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	- Address for the TCD.

### 3.8.3.28 Function Mcl\_DmaTcdSetDaddr

Mcl\_DmaTcdSetDaddr.

**Details:**

This function is used for setting the DADDR for a TCD based on the address of the TCD.

**Return:** .

**Implements:** Mcl\_DmaTcdSetDaddr\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** `void Mcl_DmaTcdSetDaddr(Mcl_DmaTcdType *pTcdAddress, uint32 u32Daddr);`

**Table 3-88. Mcl\_DmaTcdSetDaddr Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	- Address for the TCD.
uint32	u32Daddr	input	- Destination Address.

### 3.8.3.29 Function Mcl\_DmaTcdSetDlast

Mcl\_DmaTcdSetDlast.

**Details:**

This function is used for setting the DLAST for a TCD, when Enable Scatter Gather is not set, based on the address of the TCD.

**Return:** .

**Implements:** Mcl\_DmaTcdSetDlast\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaTcdSetDlast(Mcl\_DmaTcdType \*pTcdAddress, sint32 s32Dlast);

**Table 3-89. Mcl\_DmaTcdSetDlast Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	- Address for the TCD.
sint32	s32Dlast	input	- Adjustment value added to the destination address at the completion of the major iteration count.

### 3.8.3.30 Function Mcl\_DmaTcdSetDModuloAndSize

Mcl\_DmaTcdSetDModuloAndSize.

**Details:**

This function is used for setting the DMOD and DSIZE for a TCD based on the address of the TCD. SMOD and SSIZE will be preserved.

**Return:** .

**Implements:** Mcl\_DmaTcdSetDModuloAndSize\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaTcdSetDModuloAndSize(Mcl\_DmaTcdType \*pTcdAddress, uint8 u8DModulo, Mcl\_DmaSizeType DSize);

**Table 3-90. Mcl\_DmaTcdSetDModuloAndSize Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	- Address for the TCD.
uint8	u8DModulo	input	- Destination Address Modulo.
Mcl_DmaSizeType	DSize	input	- Destination data transfer size.

### 3.8.3.31 Function Mcl\_DmaTcdSetDoff

Mcl\_DmaTcdSetDoff.

**Details:**

This function is used for setting the Destination offset for a TCD based on the address of the TCD.

**Return:** .

**Implements:** Mcl\_DmaTcdSetDoff\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaTcdSetDoff(Mcl\_DmaTcdType \*pTcdAddress, sint16 s16Doff);

**Table 3-91. Mcl\_DmaTcdSetDoff Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	- Address for the TCD.
sint16	s16Doff	input	- Destination Address Offset.

### 3.8.3.32 Function Mcl\_DmaTcdSetFlags

Mcl\_DmaTcdSetFlags.

**Details:**

This function is used for setting the Channel Done, Channel Active, Enable channel-to-channel linking on major loop complete, Enable Scatter Gather Processing, Disable Request, Enable an interrupt when major counter is half complete, Enable an interrupt when major iteration count completes, Channel Start flags for a TCD based on the address of the TCD.

**Return:** .

**Implements:** Mcl\_DmaTcdSetFlags\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaTcdSetFlags(Mcl\_DmaTcdType \*pTcdAddress, uint8 u8Flags);

**Table 3-92. Mcl\_DmaTcdSetFlags Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	- Address for the TCD.
uint8	u8Flags	input	- Flags to be set.

### 3.8.3.33 Function Mcl\_DmaTcdSetIntMaj

Mcl\_DmaTcdSetIntMaj.

**Details:**

This function enables the interrupts when major iteration count completes for a TCD based on the address of the TCD.

**Return:** .

**Implements:** Mcl\_DmaTcdSetIntMaj\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaTcdSetIntMaj (Mcl\_DmaTcdType \*pTcdAddress);

**Table 3-93. Mcl\_DmaTcdSetIntMaj Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	- Address for the TCD.

### 3.8.3.34 Function Mcl\_DmaTcdSetIterCount

Mcl\_DmaTcdSetIterCount.

**Details:**

This function is used for setting the major iteration count (CITER and BITER fields) for a TCD based on the address of the TCD.

**Return:** .

**Implements:** Mcl\_DmaTcdSetIterCount\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaTcdSetIterCount (Mcl\_DmaTcdType \*pTcdAddress, uint16 u16Iter);

**Table 3-94. Mcl\_DmaTcdSetIterCount Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	- Address for the TCD.
uint16	u16Iter	input	- Value for major iteration count.

### 3.8.3.35 Function Mcl\_DmaTcdSetLinkAndIterCount

Mcl\_DmaTcdSetLinkAndIterCount.

#### **Details:**

This function is used for enabling channel-to-channel linking (ELINK field set), setting the linked channel number (LINKCH field) and the major iteration count (CITER & BITER fields) for a TCD based on the address of the TCD.

#### **Return:** .

**Implements:** Mcl\_DmaTcdSetLinkAndIterCount\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaTcdSetLinkAndIterCount(Mcl\_DmaTcdType \*pTcdAddress, Mcl\_ChannelType LinkCh, uint16 u16Iter);

**Table 3-95. Mcl\_DmaTcdSetLinkAndIterCount Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	- Address for the TCD.
Mcl_ChannelType	LinkCh	input	- Linked DMA channel number.
uint16	u16Iter	input	- Is the value for major iteration count.

### 3.8.3.36 Function Mcl\_DmaTcdSetMinorLoop

Mcl\_DmaTcdSetMinorLoop.

#### **Details:**



This function is used for setting the Smloe, Dmloe, Mloff and NBytes for minor loop offset when minor loop is enabled, for a TCD based on the address of the TCD. If offset is disabled (Smloe = FALSE and Dmloe = FALSE), the values set will be Smloe, Dmloe and NBytes for the rest of the register. If offset is enabled (Smloe = TRUE or Dmloe = true) the values set will be Smloe, Dmloe, Mloff, Nbytes.

**Return:** .

**Implements:** Mcl\_DmaTcdSetMinorLoop\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaTcdSetMinorLoop(Mcl\_DmaTcdType \*pTcdAddress, boolean bSmloe, boolean bDmloe, sint32 s32Mloff, uint32 u32NBytes);

**Table 3-96. Mcl\_DmaTcdSetMinorLoop Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	- Address for the TCD.
boolean	bSmloe	input	- Source minor loop offset enable.
boolean	bDmloe	input	- Destination minor loop offset enable.
sint32	s32Mloff	input	- Offset applied to the source or destination address.
uint32	u32NBytes	input	- Minor Byte Transfer Count.

### 3.8.3.37 Function Mcl\_DmaTcdSetSaddr

Mcl\_DmaTcdSetSaddr.

**Details:**

This function is used for setting the SADDR for a TCD based on the address of the TCD.

**Return:** .

**Implements:** Mcl\_DmaTcdSetSaddr\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaTcdSetSaddr(Mcl\_DmaTcdType \*pTcdAddress, uint32 u32Saddr);

**Table 3-97. Mcl\_DmaTcdSetSaddr Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	- Address for the TCD.

*Table continues on the next page...*

**Table 3-97. Mcl\_DmaTcdSetSaddr Arguments (continued)**

Type	Name	Direction	Description
	Saddr	input	- Address to set in SADDR.

### 3.8.3.38 Function Mcl\_DmaTcdSetSga

Mcl\_DmaTcdSetSga.

**Details:**

This function is used for setting the SGA for a TCD, when Enable Scatter Gather is set, based on the address of the TCD.

**Return:** .

**Implements:** Mcl\_DmaTcdSetSga\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaTcdSetSga(Mcl\_DmaTcdType \*pTcdAddress, uint32 u32Sga);

**Table 3-98. Mcl\_DmaTcdSetSga Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	- Address for the TCD.
uint32	u32Sga	input	- This address points to the beginning of a region containing the next TCD to be loaded into this channel.

### 3.8.3.39 Function Mcl\_DmaTcdSetSlast

Mcl\_DmaTcdSetSlast.

**Details:**

This function is used for setting the SLAST for a TCD based on the address of the TCD.

**Return:** .

**Implements:** Mcl\_DmaTcdSetSlast\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaTcdSetSlast(Mcl\_DmaTcdType \*pTcdAddress, sint32 s32Slast);

**Table 3-99. Mcl\_DmaTcdSetSlast Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	- Address for the TCD.
sint32	s32Slast	input	- Last Source Address Adjustment.

### 3.8.3.40 Function Mcl\_DmaTcdSetSModuloAndSize

Mcl\_DmaTcdSetSModuloAndSize.

**Details:**

This function is used for setting the SMOD and SSize for a TCD based on the address of the TCD. DMOD and DSize will be preserved.

**Return:** .

**Implements:** Mcl\_DmaTcdSetSModuloAndSize\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaTcdSetSModuloAndSize(Mcl\_DmaTcdType \*pTcdAddress, uint8 u8SModulo, Mcl\_DmaSizeType SSize);

**Table 3-100. Mcl\_DmaTcdSetSModuloAndSize Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	- Address for the TCD.
uint8	u8SModulo	input	- Source Address Modulo.
Mcl_DmaSizeType	SSize	input	- Source data transfer size.

### 3.8.3.41 Function Mcl\_DmaTcdSetSoff

Mcl\_DmaTcdSetSoff.

**Details:**

This function is used for setting the SOFF for a TCD based on the address of the TCD.

**Return:** .

**Implements:** Mcl\_DmaTcdSetSoff\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaTcdSetSoff(Mcl\_DmaTcdType \*pTcdAddress, sint16 s16Soff);

**Table 3-101. Mcl\_DmaTcdSetSoff Arguments**

Type	Name	Direction	Description
Mcl_DmaTcdType*	pTcdAddress	input	- Address for the TCD.
sint16	s16Soff	input	- Source address offset.

### 3.8.3.42 Function Mcl\_DmaUpdateDestAddress

Mcl\_UpdateDmaDestAddress.

**Details:**

This function is used for updating the destination address.

**Return:** void.

**Implements:** Mcl\_DmaUpdateDestAddress\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaUpdateDestAddress(Mcl\_ChannelType ChannelNumber, uint32 daddr);

**Table 3-102. Mcl\_DmaUpdateDestAddress Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	ChannelNumber	input	- Mcl Channel. daddr - Destination address.

### 3.8.3.43 Function Mcl\_DmaUpdateIterCount

Mcl\_DmaUpdateIterCount.

**Details:**

This function is used for updating the iteration count bits.

**Return:** void.

**Implements:** Mcl\_DmaUpdateIterCount\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DmaUpdateIterCount (Mcl\_ChannelType ChannelNumber, uint16 u16Iter);

**Table 3-103. Mcl\_DmaUpdateIterCount Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	ChannelNumber	input	- Mcl Channel for updating the iteration count. u16Iter - iteration number.

### 3.8.3.44 Function Mcl\_DmaGetCrtIterCount

Mcl\_DmaGetCrtIterCount.

**Details:**

This function is used for getting the current iteration count for a specified channel.

**Return:** iteration number.

**Implements:** Mcl\_DmaGetCrtIterCount\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** uint16 Mcl\_DmaGetCrtIterCount (Mcl\_ChannelType ChannelNumber);

**Table 3-104. Mcl\_DmaGetCrtIterCount Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	ChannelNumber	input	- Mcl Channel.

### 3.8.3.45 Function Mcl\_DmaGetStartIterCount

Mcl\_DmaGetStartIterCount.

**Details:**

This function is used for getting the staring iteration count for a specified channel.

**Return:** iteration number.

**Implements:** Mcl\_DmaGetStartIterCount\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** `uint16 Mcl_DmaGetStartIterCount(Mcl_ChannelType ChannelNumber);`

**Table 3-105. Mcl\_DmaGetStartIterCount Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	ChannelNumber	input	- Mcl Channel.

### 3.8.3.46 Function Mcl\_Init

This function initializes the Dma driver.

**Details:**

This service is a non reentrant function used for driver initialization. The Initialization function shall initialize all relevant registers of the configured hardware with the values of the structure referenced by the parameter ConfigPtr. Mcl\_Init shall always have a valid pointer as a parameter except for the Variant PC if only one configuration set is used, in which case a NULL pointer shall be passed to the initialization function.

**Return:** void.

**Implements:** Mcl\_Init\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** `void Mcl_Init(const Mcl_ConfigType *ConfigPtr);`

**Table 3-106. Mcl\_Init Arguments**

Type	Name	Direction	Description
constMcl_ConfigType*	ConfigPtr	input	Pointer to a selected configuration structure.

### 3.8.3.47 Function Mcl\_DeInit

This function initializes the Dma driver.

#### Details:

This service is a non reentrant function used for driver initialization. This function de-initializes the Mcl driver. Returns all underlying hardware to a state comparable to their power on reset state, and de-initialize the MCL driver.

**Return:** void.

**Implements:** Mcl\_DeInit\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_DeInit(void);

**Table 3-107. Mcl\_DeInit Arguments**

Type	Name	Direction	Description
void		input	This function has no argument

### 3.8.3.48 Function Mcl\_DmaGetGlobalErrorStatus

Mcl\_DmaGetGlobalErrorStatus.

#### Details:

This function is used for getting the DMA instance global error status provided by hardware.

**Return:** void .

**Implements:** Mcl\_DmaGetGlobalErrorStatus\_Activity

**Prototype:** void Mcl\_DmaGetGlobalErrorStatus(Mcl\_DmaInstanceType dmaInstance, Mcl\_DmaGlobalErrorStatusType\* dmaGlobalErrorStatus);

**Table 3-108. Mcl\_DmaGetGlobalErrorStatus Arguments**

Type	Name	Direction	Description
Mcl_DmaInstanceType	dmaInstance	input	- DMA instance identifier.

*Table continues on the next page...*

**Table 3-108. Mcl\_DmaGetGlobalErrorStatus Arguments (continued)**

Type	Name	Direction	Description
Mcl_DmaGlobalErrorStatusType*	dmaGlobalErrorStatus	input	- pointer to the error information.
Mcl_DmaGlobalErrorStatusType*	dmaGlobalErrorStatus	output	- pointer to the error information.

### 3.8.3.49 Function Mcl\_DmaGetChannelErrorStatus

Mcl\_DmaGetChannelErrorStatus.

#### **Details:**

This function is used for getting the physical DMA channel for a given Mcl channel.

**Return:** Mcl\_DmaChannelErrorStatusType - provides the error information for a specified logical channel .

**Implements:** Mcl\_DmaGetChannelErrorStatus\_Activity

**Prototype:** Mcl\_DmaChannelErrorStatusType Mcl\_DmaGetChannelErrorStatus(Mcl\_ChannelType logicalChannel);

**Table 3-109. Mcl\_DmaGetChannelErrorStatus Arguments**

Type	Name	Direction	Description
Mcl_ChannelType	logicalChannel	input	- MCL logical channel.

### 3.8.3.50 Function Mcl\_GetVersionInfo

This service returns the version information of this module.

#### **Details:**

This service is Non reentrant and returns the version information of this module. The version information includes:

- Module Id



- Vendor Id
- Vendor specific version numbers If source code for caller and callee of this function is available this function should be realized as a macro. The macro should be defined in the modules header file.

**Return:** void .

**Implements:** Mcl\_GetVersionInfo\_Activity

**Violates:** Violates MISRA 2004 Required Rule 8.10 could be made static

**Prototype:** void Mcl\_GetVersionInfo(Std\_VersionInfoType \*versioninfo);

**Table 3-110. Mcl\_GetVersionInfo Arguments**

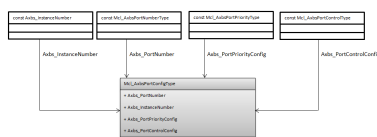
Type	Name	Direction	Description
Std_VersionInfoType *	versioninfo	output	Pointer to location to store version info .

## 3.8.4 Structs Reference

This chapter describes the structs supported by the MCL driver.

### 3.8.4.1 Structure Mcl\_AxbsPortConfigType

Mcl Dma high level configuration structure.



**Figure 3-1. Struct Mcl\_AxbsPortConfigType**

**Implements:** Mcl\_AxbsPortConfigType

**Declaration:**

```

typedef struct
{
    const Mcl_AxbsPortNumberType Axbs_PortNumber,
    const Axbs_InstanceNumber Axbs_InstanceNumber,
    const Mcl_AxbsPortPriorityType Axbs_PortPriorityConfig,
    const Mcl_AxbsPortControlType Axbs_PortControlConfig
} Mcl_AxbsPortConfigType;
  
```

Table 3-111. Structure Mcl\_AxbsPortConfigType member description

Member	Description
Axbs_PortNumber	Hardware port number.
Axbs_InstanceNumber	Hardware instance.
Axbs_PortPriorityConfig	Port priority config.
Axbs_PortControlConfig	Port control config.

3.8.4.2 Structure Mcl\_ChannelConfigType

Mcl Dma channel high level configuration structure.

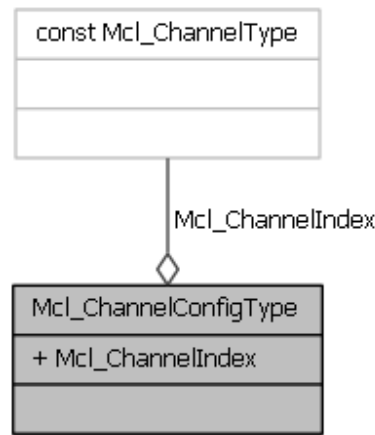


Figure 3-2. Struct Mcl\_ChannelConfigType

**Implements:** Mcl\_Dma\_ChannelConfigType\_struct

**Declaration:**

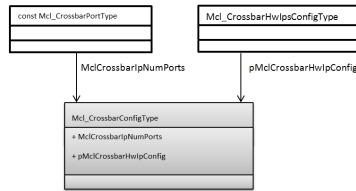
```
typedef struct
{
    const Mcl_ChannelType Mcl_ChannelIndex
} Mcl_ChannelConfigType;
```

Table 3-112. Structure Mcl\_ChannelConfigType member description

Member	Description
Mcl_ChannelIndex	Dma Channel configuration.

### 3.8.4.3 Structure Mcl\_CrossbarConfigType

Mcl Dma high level configuration structure.



**Figure 3-3. Struct Mcl\_CrossbarConfigType**

**Implements:** Mcl\_CrossbarConfigType

#### Declaration:

```

typedef struct
{
    const Mcl_CrossbarPortType MclCrossbarIpNumPorts,
    const Mcl_CrossbarHwIpsConfigType *const pMclCrossbarHwIpConfig
} Mcl_CrossbarConfigType;
  
```

**Table 3-113. Structure Mcl\_CrossbarConfigType member description**

Member	Description
MclCrossbarIpNumPorts	The number of configured channels.
pMclCrossbarHwIpConfig	configuration of the crossbar IP.

### 3.8.4.4 Structure Mcl\_DmaInitConfigType

Mcl Dma high level configuration structure.

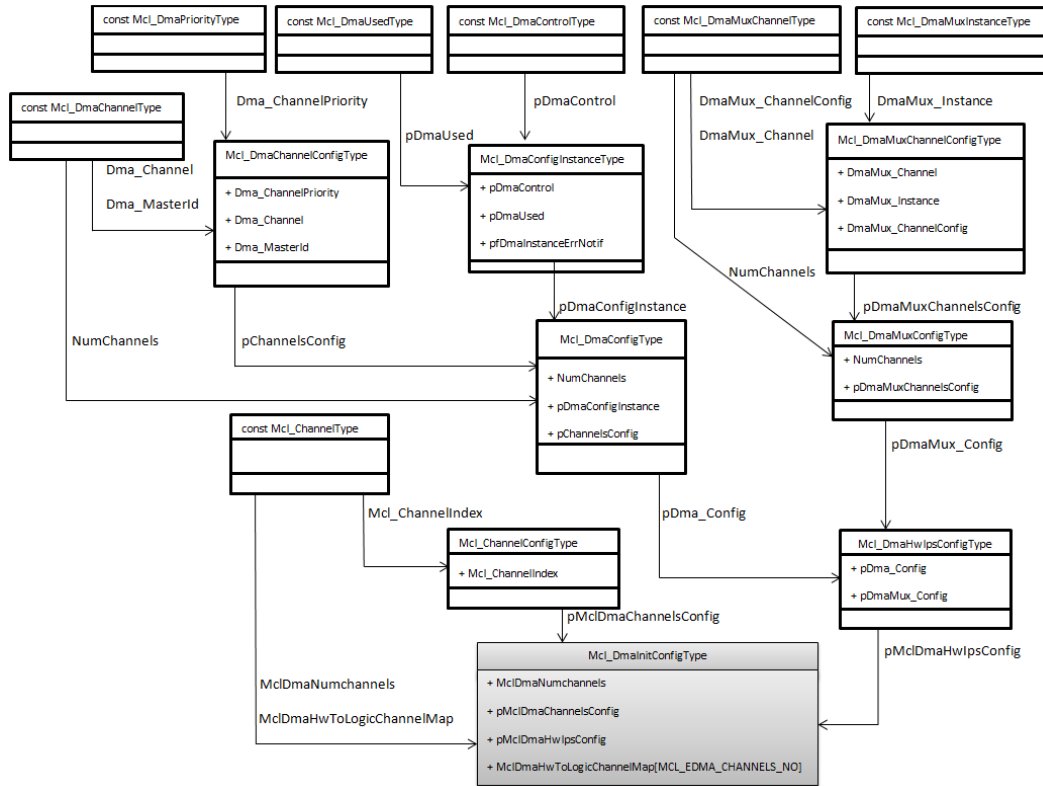


Figure 3-4. Struct Mcl\_DmaInitConfigType

**Implements:** Mcl\_DmaInitConfigType\_struct**Declaration:**

```

typedef struct
{
    const Mcl_ChannelType MclDmaNumchannels,
    const Mcl_ChannelConfigType *const pMclDmaChannelsConfig,
    const Mcl_DmaHwIpsConfigType * pMclDmaHwIpsConfig,
    #if ((MCL_DMA_NOTIFICATION_SUPPORTED==STD_ON) || \
        (MCL_DMA_GET_GLOBAL_ERR_STATUS_API==STD_ON))
    const Mcl_ChannelType MclDmaHwToLogicChannelMap[MCL_EDMA_CHANNELS_NO]
    #endif
} Mcl_DmaInitConfigType;

```

**Table 3-114. Structure Mcl\_DmaInitConfigType member description**

Member	Description
MclDmaNumchannels	Number of channels in the Mcl configuration.
pMclDmaChannelsConfig	Pointer to the list of Dma configured channels.
pMclDmaHwIpsConfig	IPs data generic configuration.
MclDmaHwToLogicChannelMap[MCL_EDMA_CHANNELS_NO]	Index table to translate eDma HW channels on to logical channels used to process interrupts for notifications.

### 3.8.4.5 Structure Mcl\_ConfigType

Mcl high level configuration structure.

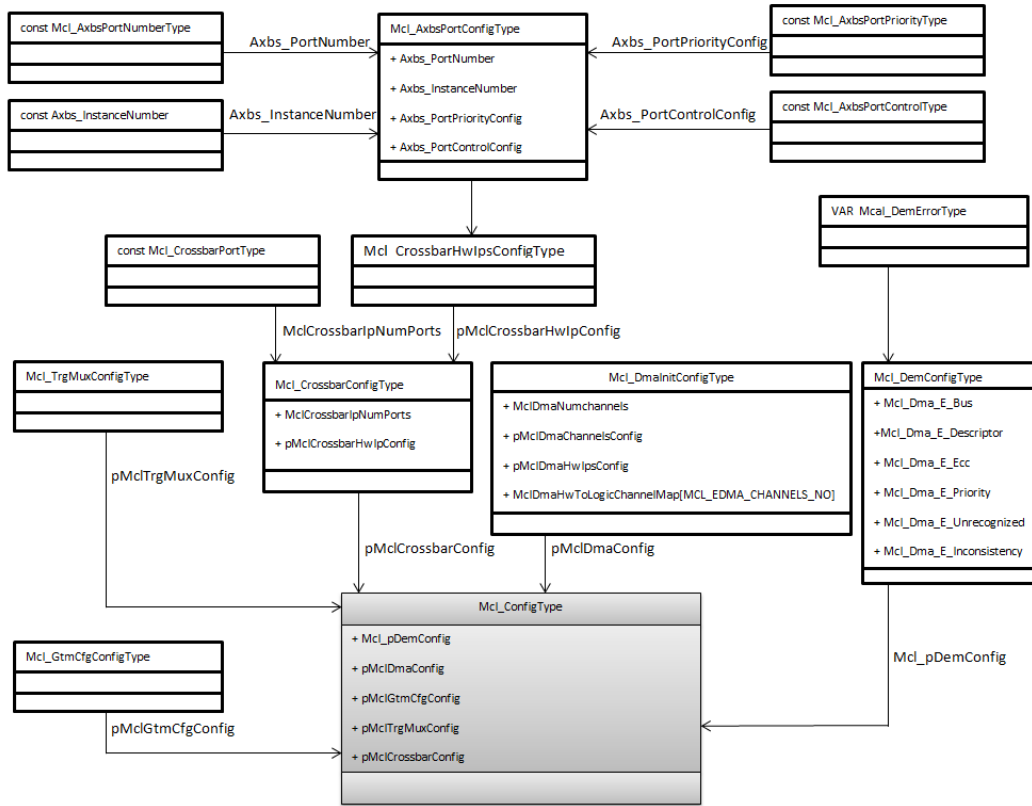


Figure 3-5. Struct Mcl\_ConfigType

**Implements:** Mcl\_ConfigType\_struct

**Declaration:**

```
typedef struct
```

```

{
    #if (MCL_ENABLE_DMA == STD_ON)
    #if (MCL_DISABLE_DEM_REPORT_ERROR_STATUS == STD_OFF)
        const Mcl_DemConfigType * Mcl_pDemConfig,
    #endif
        const Mcl_DmaInitConfigType * pMclDmaConfig,
    #endif
    #if (MCL_ENABLE_GTMCFG == STD_ON)
        const Mcl_GtmCfgConfigType * pMclGtmCfgConfig,
    #endif
    #if (MCL_ENABLE_TRGMUX == STD_ON)
        const Mcl_TrngMuxConfigType * pMclTrngMuxConfig,
    #endif
    #if (MCL_ENABLE_CROSSBAR == STD_ON)
        const Mcl_CrossbarConfigType * pMclCrossbarConfig
    #endif
} Mcl_ConfigType;
```

Table 3-115. Structure Mcl\_ConfigType member description

Member	Description
Mcl_pDemConfig	DEM error reporting configuration.
pMclDmaConfig	
pMclGtmCfgConfig	Pointer to the GTMCFG configuration.
pMclTrgMuxConfig	Pointer to the TrgMuxCFG configuration.
pMclCrossbarConfig	Pointer to the Crossbar configuration.

3.8.4.6 Structure Mcl\_DmaChannelConfigType

Dma channel configuration structure.

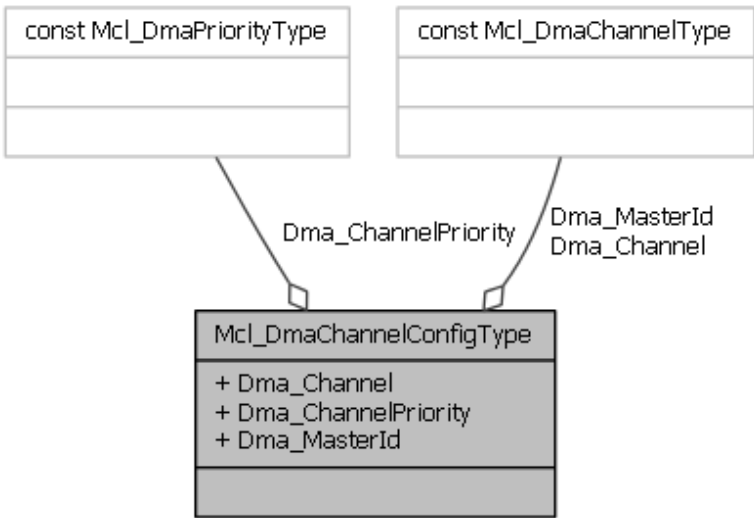


Figure 3-6. Struct Mcl\_DmaChannelConfigType

**Implements:** Mcl\_DmaChannelConfigType\_struct

**Declaration:**

```
typedef struct
{
    const Mcl_DmaChannelType Dma_Channel,
    const Mcl_DmaPriorityType Dma_ChannelPriority,
    const Mcl_DmaChannelType Dma_MasterId
} Mcl_DmaChannelConfigType;
```

Table 3-116. Structure Mcl\_DmaChannelConfigType member description

Member	Description
Dma_Channel	eDma channel used

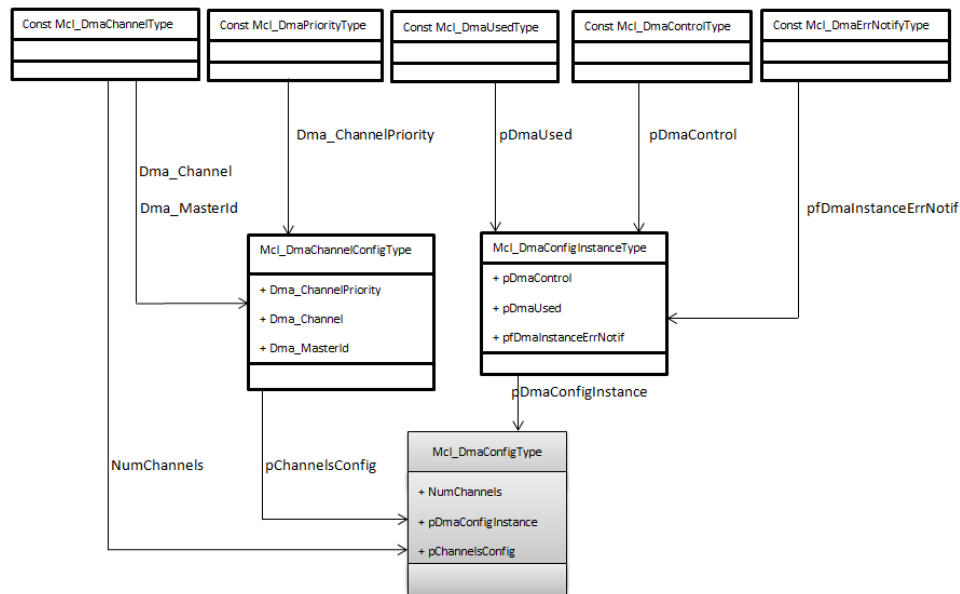
Table continues on the next page...

**Table 3-116. Structure Mcl\_DmaChannelConfigType member description (continued)**

Member	Description
Dma_ChannelPriority	Channel ECP, DPA and Priority.
Dma_MasterId	eDma channel master ID replication

### 3.8.4.7 Structure Mcl\_DmaConfigType

Dma configuration structure.

**Figure 3-7. Struct Mcl\_DmaConfigType**

**Implements:** `Mcl_DmaConfigType_struct`

**Declaration:**

```
typedef struct
{
    const Mcl_DmaChannelType NumChannels,
    const Mcl_DmaConfigInstanceType*const pDmaConfigInstance,
    const Mcl_DmaChannelConfigType *const pChannelsConfig
} Mcl_DmaConfigType;
```

**Table 3-117. Structure Mcl\_DmaConfigType member description**

Member	Description
NumChannels	Number of eDma channels in the Mcl configuration.
pDmaConfigInstance	Pointer to the configured channels for eDma.
pChannelsConfig	Pointer to the configured channels for eDma.

### 3.8.4.8 Structure Mcl\_DmaMuxChannelConfigType

DmaMux channel configuration structure.

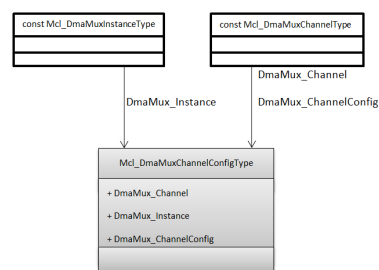


Figure 3-8. Struct Mcl\_DmaMuxChannelConfigType

**Implements:** DmaMux\_ChannelConfigType\_struct

**Declaration:**

```
typedef struct
{
    const Mcl_DmaMuxChannelType DmaMux_Channel,
    const Mcl_DmaMuxInstanceType DmaMux_Instance,
    const Mcl_DmaMuxChannelType DmaMux_ChannelConfig
} Mcl_DmaMuxChannelConfigType;
```

Table 3-118. Structure Mcl\_DmaMuxChannelConfigType member description

Member	Description
DmaMux_Channel	eDma channel used
DmaMux_Instance	DmaMux instance used
DmaMux_ChannelConfig	Channel Enable, Trig and Source.

### 3.8.4.9 Structure Mcl\_DmaMuxConfigType

DmaMux configuration structure.



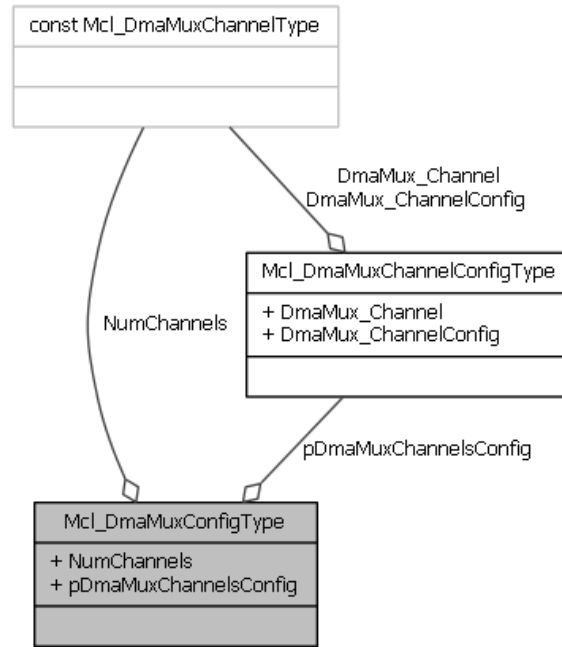


Figure 3-9. Struct Mcl\_DmaMuxConfigType

**Implements:** DmaMux\_ConfigType\_struct

### Declaration:

```

typedef struct
{
    const Mcl_DmaMuxChannelType NumChannels,
    const Mcl_DmaMuxChannelConfigType *const
pDmaMuxChannelsConfig
} Mcl_DmaMuxConfigType;
  
```

Table 3-119. Structure Mcl\_DmaMuxConfigType member description

Member	Description
NumChannels	Number of DmaMux configured channels.
pDmaMuxChannelsConfig	Pointer to the list of Dma configured channels.

### 3.8.4.10 Structure Mcl\_DmaTcdAttributesType

structure used for a basic configuration of a TCD

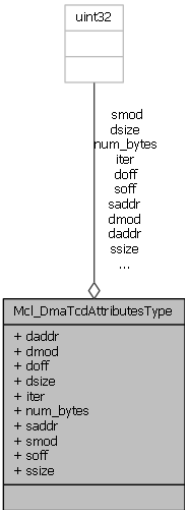


Figure 3-10. Struct Mcl\_DmaTcdAttributesType

**Implements:** Mcl\_DmaTcdAttributesType\_struct

**Declaration:**

```
typedef struct
{
    uint32 saddr,
    uint32 daddr,

    uint32 ssize,
    uint32 dsize,

    uint32 soff,
    uint32 doff,
    uint32 smod,
    uint32 dmod,
    uint32 num_bytes,
    uint32 iter
} Mcl_DmaTcdAttributesType;
```

Table 3-120. Structure Mcl\_DmaTcdAttributesType member description

Member	Description
saddr	source address
daddr	destination address
ssize	source transfer size
dsize	destination transfer size
soff	source address offset
doff	destination address offset
smod	source address modulo
dmod	destination address modulo

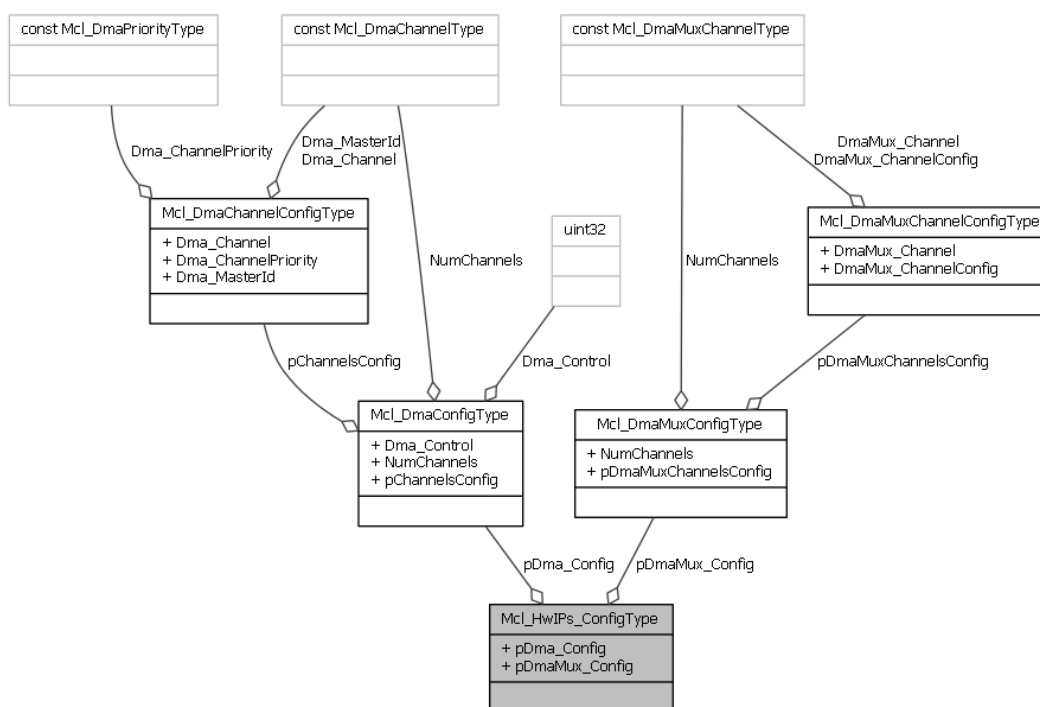
Table continues on the next page...

**Table 3-120. Structure Mcl\_DmaTcdAttributesType member description (continued)**

Member	Description
num_bytes	number of bytes to be transferred
iter	iteration count

### 3.8.4.11 Structure Mcl\_DmaHwIpsConfigType

Mcl driver configuration structure.

**Figure 3-11. Struct Mcl\_DmaHwIpsConfigType**

#### Details:

Configuration for DMA\_MUX and eDMA modules. Used by "Mcl\_ConfigType" structure.

#### **Declaration:**

```
typedef struct
{
    const Mcl_DmaConfigType * pDma_Config,
    const Mcl_DmaMuxConfigType * pDmaMux_Config
} Mcl_DmaHwIpsConfigType;
```

Table 3-121. Structure Mcl\_DmaHwIpsConfigType member description

Member	Description
pDma_Config	Configuration for eDMA (Enhanced Direct Memory Access) hardware IP.
pDmaMux_Config	Configuration for DMA_MUX (eDMA Channel Mux) hardware IP.

3.8.4.12 Structure Mcl\_DmaGlobalErrorStatusType

Dma channel configuration structure.

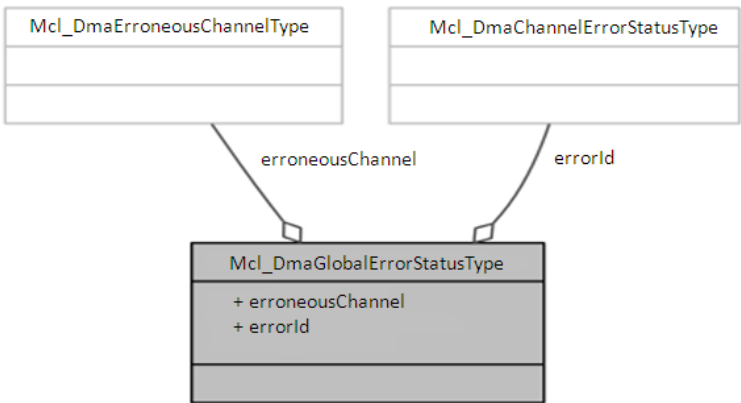


Figure 3-12. Struct Mcl\_DmaGlobalErrorStatusType

**Implements:** Mcl\_DmaGlobalErrorStatusType\_struct

**Declaration:**

```
typedef struct
{
    Mcl_DmaErroneousChannelType erroneousChannel,
    Mcl_DmaChannelErrorStatusType errorId
} Mcl_DmaGlobalErrorStatusType;
```

Table 3-122. Structure Mcl\_DmaGlobalErrorStatusType member description

Member	Description
erroneousChannel	The logic channel occurs error
errorId	Provides the numeric ID of a Mcl DMA error.

3.8.5 Types Reference

This chapter describes the type definitions supported by the MCL driver.

### 3.8.5.1 Typedef Mcl\_DmaChannelType

This gives the numeric ID (hardware channel number) of an DMA channel.

**Implements:** Mcl\_DmaChannelType\_typedef

**Type:** uint8

### 3.8.5.2 Typedef Mcl\_DmaControlType

The Dma\_ControlType contains DMA CR configuration.

**Implements:** Mcl\_DmaControlType\_typedef

**Type:** uint32

### 3.8.5.3 Typedef Mcl\_DmaPriorityType

Type for specifying the DMA channel's priority.

**Implements:** Mcl\_DmaPriorityType\_typedef

**Type:** uint16

### 3.8.5.4 Typedef Mcl\_DmaTcdType

The Dma\_TcdType contains combined bit fields for the channel's TCD.

**Implements:** Mcl\_DmaTcdType\_typedef

**Type:** uint32

### 3.8.5.5 Typedef Mcl\_DmaMuxChannelType

DmaMux channel type.

**Implements:** DmaMux\_ChannelType\_struct

**Type:** uint8

### 3.8.5.6 Typedef Mcl\_ChannelType

This gives the numeric ID of a Mcl logic channel.

For S32R274, the Mcl Logic channels are:

```
#define MCL_DMA_LOGICAL_CHANNEL_0      (0U)
#define MCL_DMA_LOGICAL_CHANNEL_1      (1U)
#define MCL_DMA_LOGICAL_CHANNEL_2      (2U)
#define MCL_DMA_LOGICAL_CHANNEL_3      (3U)
#define MCL_DMA_LOGICAL_CHANNEL_4      (4U)
#define MCL_DMA_LOGICAL_CHANNEL_5      (5U)
#define MCL_DMA_LOGICAL_CHANNEL_6      (6U)
#define MCL_DMA_LOGICAL_CHANNEL_7      (7U)
#define MCL_DMA_LOGICAL_CHANNEL_8      (8U)
#define MCL_DMA_LOGICAL_CHANNEL_9      (9U)
#define MCL_DMA_LOGICAL_CHANNEL_10     (10U)
#define MCL_DMA_LOGICAL_CHANNEL_11     (11U)
#define MCL_DMA_LOGICAL_CHANNEL_12     (12U)
#define MCL_DMA_LOGICAL_CHANNEL_13     (13U)
#define MCL_DMA_LOGICAL_CHANNEL_14     (14U)
#define MCL_DMA_LOGICAL_CHANNEL_15     (15U)
#define MCL_DMA_LOGICAL_CHANNEL_16     (16U)
#define MCL_DMA_LOGICAL_CHANNEL_17     (17U)
#define MCL_DMA_LOGICAL_CHANNEL_18     (18U)
#define MCL_DMA_LOGICAL_CHANNEL_19     (19U)
#define MCL_DMA_LOGICAL_CHANNEL_20     (20U)
#define MCL_DMA_LOGICAL_CHANNEL_21     (21U)
#define MCL_DMA_LOGICAL_CHANNEL_22     (22U)
#define MCL_DMA_LOGICAL_CHANNEL_23     (23U)
#define MCL_DMA_LOGICAL_CHANNEL_24     (24U)
#define MCL_DMA_LOGICAL_CHANNEL_25     (25U)
#define MCL_DMA_LOGICAL_CHANNEL_26     (26U)
#define MCL_DMA_LOGICAL_CHANNEL_27     (27U)
#define MCL_DMA_LOGICAL_CHANNEL_28     (28U)
#define MCL_DMA_LOGICAL_CHANNEL_29     (29U)
#define MCL_DMA_LOGICAL_CHANNEL_30     (30U)
#define MCL_DMA_LOGICAL_CHANNEL_31     (31U)
```

**Implements:** Mcl\_ChannelType\_typedef

**Type:** uint8

### 3.8.5.7 Typedef Mcl\_NotifyType

The notification functions shall have no parameters and no return value.

**Implements:** Mcl\_NotifyType\_typedef

**Type:** void(\*)

### 3.8.5.8 Typedef Mcl\_DmaInstanceType

The Mcl\_DmaInstanceType contains the DMA instance logical names. For S32R274 there is only one DMA instance(DMA\_INSTANCE0).

**Implements:** Mcl\_DmaInstanceType\_typedef

**Type:** uint8

### 3.8.5.9 Typedef Mcl\_DmaErroneousChannelType

Mcl\_DmaErroneousChannelType the numeric ID of a Mcl logic channel.

**Implements:** Mcl\_DmaErroneousChannelType\_typedef

**Type:** uint8

### 3.8.5.10 Typedef Axbs\_InstanceNumber

Axbs instance number type.

**Implements:** Axbs\_InstanceNumber\_typedef

**Type:** uint8

### 3.8.5.11 Typedef Mcl\_AxbsPortControlType

Axbs port control type.

**Implements:** Mcl\_AxbsPortControlType\_typedef

**Type:** uint32

### **3.8.5.12 Typedef Mcl\_AxbsPortNumberType**

Axbs port number type.

**Implements:** Mcl\_AxbsPortNumberType\_typedef

**Type:** uint8

### **3.8.5.13 Typedef Mcl\_AxbsPortPriorityType**

Axbs port priority config.

**Implements:** Mcl\_AxbsPortPriorityType\_typedef

**Type:** uint32

### **3.8.5.14 Typedef Mcl\_CrossbarHwIpsConfigType**

Mcl Crossbar Hw IP Configuration.

**Implements:** Mcl\_CrossbarHwIpsConfigType\_typedef

**Type:** Mcl\_AxbsPortConfigType

### **3.8.5.15 Typedef Mcl\_CrossbarPortType**

This gives the numeric ID of a Mcl logical slave port for the crossbar.

**Implements:** Mcl\_CrossbarPortType\_typedef

**Type:** uint8



## Chapter 4

# Tresos Configuration Plug-in

This chapter describes the Tresos configuration plug-in for the MCL Driver. The most of the parameters are described below.

### 4.1 Configuration elements of Mcl

Included forms :

- IMPLEMENTATION\_CONFIG\_VARIANT
- MclGeneral
- MclDemEventParameterRefs
- MclConfigSet
- MclIsrAvailable
- DMAChannel
- MclCrossbarLogicalSlavePorts
- MclCrossbarInstance
- CommonPublishedInformation

Table 4-1. Revision table

Revision	Date
4.2.0	2011-11-02

### 4.2 Form IMPLEMENTATION\_CONFIG\_VARIANT

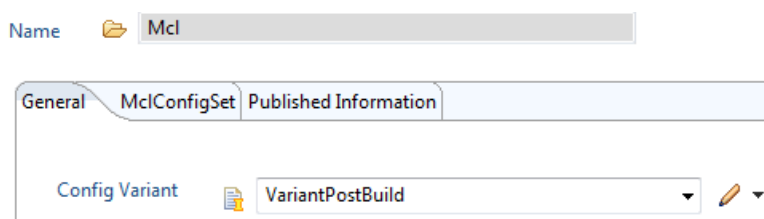
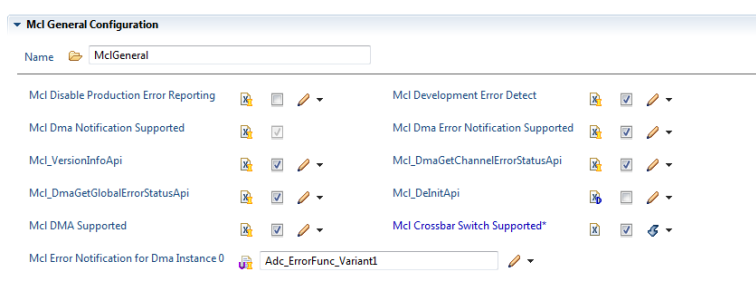


Figure 4-1. Tresos Plugin snapshot for IMPLEMENTATION\_CONFIG\_VARIANT form.

**Table 4-2. Attribute IMPLEMENTATION\_CONFIG\_VARIANT detailed description**

Property	Value
Label	Config Variant
Default	VariantPostBuild
Range	VariantPostBuild VariantPreCompile

## 4.3 Form MclGeneral

**Figure 4-2. Tresos Plugin snapshot for MclGeneral form.**

### 4.3.1 MclDisableDemReportErrorStatus (MclGeneral)

Compile switch to enable / disable Diagnostic Event Manager (DEM) for this module.

- true: Disabled.
- false: Enabled.

**Table 4-3. Attribute MclDisableDemReportErrorStatus (MclGeneral) detailed description**

Property	Value
Label	Mcl Disable Production Error Reporting
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

### 4.3.2 MclDevErrorDetect (MclGeneral)

Switches the Development Error Detection and Notification on or off.

- true: Enabled.
- false: Disabled.

**Table 4-4. Attribute MclDevErrorDetect (MclGeneral) detailed description**

Property	Value
Label	Mcl Development Error Detect
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

### 4.3.3 MclErrorChecking (MclGeneral)

Switch to indicate if the error user notification is supported.

**Table 4-5. Attribute MclErrorChecking (MclGeneral) detailed description**

Property	Value
Label	Mcl Dma Error Notification Supported
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

### 4.3.4 Mcl\_VersionInfoApi (MclGeneral)

**Table 4-6. Attribute Mcl\_VersionInfoApi (MclGeneral) detailed description**

Property	Value
Label	Mcl_VersionInfoApi
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	true

### 4.3.5 Mcl\_DmaGetChannelErrorStatusApi(MclGeneral)

Enables/Disables the get channel error status API Mcl\_DmaGetChannelErrorStatus.

**Table 4-7. Attribute Mcl\_DmaGetChannelErrorStatusApi (MclGeneral) detailed description**

Property	Value
Label	Mcl_DmaGetChannelErrorStatusApi
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

### 4.3.6 Mcl\_DmaGetGlobalErrorStatusApi(MclGeneral)

Enables/Disables the get global error status API Mcl\_DmaGetGlobalErrorStatus.

**Table 4-8. Attribute Mcl\_DmaGetGlobalErrorStatusApi (MclGeneral) detailed description**

Property	Value
Label	Mcl_DmaGetGlobalErrorStatusApi
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

### 4.3.7 EnableDMA (MclGeneral)

Enables/Disables the get global error status API Mcl\_DmaGetGlobalErrorStatus.

**Table 4-9. Attribute EnableDMA (MclGeneral) detailed description**

Property	Value
Label	Mcl DMA Supported
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

### 4.3.8 MclEnableCrossbarSwitch (MclGeneral)

Activates/Deactivates Crossbar Switch configuration.

**Table 4-10. Attribute McIEnableCrossbarSwitch (McIGeneral) detailed description**

Property	Value
Label	McI Crossbar Switch Supported
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

### 4.3.9 McIErrorNotificationDma0 (McIGeneral)

User callback function

**NOTE:** please use NULL or NULL\_PTR w/o any quotes. If the used string is different from NULL or NULL\_PTR it will be used as the configured function name.

**Table 4-11. Attribute McIErrorNotificationDma0 (McIGeneral) detailed description**

Property	Value
Label	McI Error Notification for Dma Instance 0
Type	FUNCTION-NAME
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	NULL_PTR

## 4.4 Form McIDemEventParameterRefs

Container for the references to DemEventParameter elements which shall be invoked using the Dem function in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter/DemEventId value.

Name	Value
MCL_DMA_E_DESCRIPTOR	/Dem/Dem/DemConfigSet_0/MCL_DMA_E_DESCRIPTOR
MCL_DMA_E_ECC	
MCL_DMA_E_BUS	/Dem/Dem/DemConfigSet_0/MCL_DMA_E_ECC
MCL_DMA_E_PRIORITY	/Dem/Dem/DemConfigSet_0/MCL_DMA_E_PRIORITY
MCL_DMA_E_INCONSISTENCY	
MCL_DMA_E_UNRECOGNIZED	

**Figure 4-3. Tresos Plugin snapshot for McIDemEventParameterRefs form.**

#### 4.4.1 MCL\_DMA\_E\_DESCRIPTOR (MclDemEventParameterRefs)

The DEM event MCL\_DMA\_E\_DESCRIPTOR is reported by the software when one of the APIs Mcl\_DmaGetGlobalErrorStatus or Mcl\_DmaGetChannelErrorStatus is called. The APIs get the error status from hardware registers. This means it will be reported asynchronous to the occurrence of the error condition in hardware. For a synchronous error reporting, the user should configure the MCL DMA error notification. Description of the error condition in the hardware: This error is reported when the DMA TCD is incorrectly configured, this means when one of the following rules is broken:

- The addresses and offsets must be aligned on 0-modulo-transfer-size boundaries.
- The minor loop byte count must be a multiple of the source and destination transfer sizes.
- All source reads and destination writes must be configured to the natural boundary of the programmed transfer size respectively.
- If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST\_SGA) is not aligned on a 32- byte boundary.
- If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCDn\_CITER[E\_LINK] bit does not equal the TCDn\_BITER[E\_LINK] bit.

If enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, are reported by the hardware as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported by the hardware when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported by the hardware when the link operation is serviced at minor loop completion.

**Table 4-12. Attribute MCL\_DMA\_E\_DESCRIPTOR (MclDemEventParameterRefs) detailed description**

Property	Value
Type	SYMBOLIC-NAME-REFERENCE
Origin	AUTOSAR_ECUC
Enable	true

#### 4.4.2 MCL\_DMA\_E\_ECC (MclDemEventParameterRefs)

**Table 4-13. Attribute MCL\_DMA\_E\_ECC (MclDemEventParameterRefs) detailed description**

Property	Value
Type	SYMBOLIC-NAME-REFERENCE
Origin	AUTOSAR_ECUC
Enable	true

### 4.4.3 MCL\_DMA\_E\_BUS (McIdemEventParameterRefs)

The DEM event MCL\_DMA\_E\_BUS is reported by the software when one of the APIs Mcl\_DmaGetGlobalErrorStatus or Mcl\_DmaGetChannelErrorStatus is called. The APIs get the error status from hardware registers. This means it will be reported asynchronous to the occurrence of the error condition in hardware. For a synchronous error reporting, the user should configure the MCL DMA error notification. Description of the error condition in the hardware:

- The error condition related to this event occurs in hardware when a source bus error or a destination bus error is reported by the DMA hardware during a transfer.
- If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and current iteration count at the point of the fault. When a system bus error occurs, the channel terminates after the next transfer. Due to pipeline effect, the next transfer is already in progress when the bus error is received by the eDMA. If a bus error occurs on the last read prior to beginning the write sequence, the write executes using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel terminates due to the destination bus error.

**Table 4-14. Attribute MCL\_DMA\_E\_BUS (McIdemEventParameterRefs) detailed description**

Property	Value
Type	SYMBOLIC-NAME-REFERENCE
Origin	AUTOSAR_ECUC
Enable	true

### 4.4.4 MCL\_DMA\_E\_PRIORITY (McIdemEventParameterRefs)

The DEM event MCL\_DMA\_E\_PRIORITY is reported by the software when one of the APIs Mcl\_DmaGetGlobalErrorStatus or Mcl\_DmaGetChannelErrorStatus is called. The APIs get the error status from hardware registers. This means it will be reported asynchronous to the occurrence of the error condition in hardware. For a synchronous error reporting, the user should configure the MCL DMA error notification. Description of the error condition in the hardware: A priority configuration error happens in the fixed arbitration mode and it is caused by any two channel priorities being equal within a group of channels.

**Table 4-15. Attribute MCL\_DMA\_E\_PRIORITY (MclDemEventParameterRefs) detailed description**

Property	Value
Type	SYMBOLIC-NAME-REFERENCE
Origin	AUTOSAR_ECUC
Enable	true

#### 4.4.5 MCL\_DMA\_E\_INCONSISTENCY (MclDemEventParameterRefs)

The DEM event MCL\_DMA\_E\_INCONSISTENCY is reported by the software when one of the APIs Mcl\_DmaGetGlobalErrorStatus or Mcl\_DmaGetChannelErrorStatus is called. The APIs get the error status from hardware registers. This means it will be reported asynchronous to the occurrence of the error condition in hardware. For a synchronous error reporting, the user should configure the MCL DMA error notification. This error event is set by software if the registers DMA\_ES and DMA\_ERR report inconsistent error information, in one of following cases: • DMA\_ES reports errors for a respective channel and DMA\_ERR reports no error for that channel • DMA\_ERR reports errors for a respective channel and DMA\_ES reports no error for that channel • DMA\_ES and DMA\_ERR report errors for different channels Error code

MCL\_DMA\_E\_INCONSISTENCY marks that the DMA might have met an error condition, but this is not clear because the hardware reports it in inconsistent matter.

**Table 4-16. Attribute MCL\_DMA\_E\_INCONSISTENCY (MclDemEventParameterRefs) detailed description**

Property	Value
Type	SYMBOLIC-NAME-REFERENCE
Origin	AUTOSAR_ECUC
Enable	true

#### 4.4.6 MCL\_DMA\_E\_UNRECOGNIZED (MclDemEventParameterRefs)

The DEM event MCL\_DMA\_E\_UNRECOGNIZED is reported by the software when one of the APIs Mcl\_DmaGetGlobalErrorStatus or Mcl\_DmaGetChannelErrorStatus is called. The APIs get the error status from hardware registers. This means it will be reported asynchronous to the occurrence of the error condition in hardware. For a synchronous error reporting, the user should configure the MCL DMA error notification.



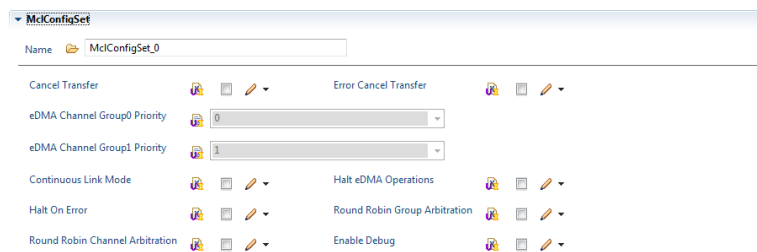
This error event is set by software if the registers DMA\_ES and DMA\_ERR report error for the same channel, but the register DMA\_ES doesn't provide any error status (no ECC error, no bus error, no descriptor error, no priority error). This might happen because of issues in the hardware.

**Table 4-17. Attribute MCL\_DMA\_E\_UNRECOGNIZED (McIdemEventParameterRefs) detailed description**

Property	Value
Type	SYMBOLIC-NAME-REFERENCE
Origin	AUTOSAR_ECUC
Enable	true

## 4.5 Form McIConfigSet

This container is the base for a multiple configuration set



**Figure 4-4. Tresos Plugin snapshot for McIConfigSet form.**

### 4.5.1 McIEDMA\_CX (McIConfigSet)

DMA\_CR[CX]. Cancel Transfer. 0 - Normal operation. 1 - Cancel the remaining data transfer. Stop the executing channel and force the minor loop to be finished. The cancel takes effect after the last write of the current read/write sequence. The CXFR bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the minor loop was completed. Note: Implementation Specific Parameter.

**Table 4-18. Attribute McIEDMA\_CX (McIConfigSet) detailed description**

Property	Value
Label	Cancel Transfer
Type	BOOLEAN
Origin	Custom

*Table continues on the next page...*

**Table 4-18. Attribute McIEDMA\_CX (MclConfigSet) detailed description (continued)**

Property	Value
Symbolic Name	false
Default	false

### 4.5.2 McIEDMA\_ECX (MclConfigSet)

DMA\_CR[ECX]. Error Cancel Transfer. 0 - Normal operation. 1 - Cancel the remaining data transfer in the same fashion as the CX cancel transfer. Stop the executing channel and force the minor loop to be finished. The cancel takes effect after the last write of the current read/write sequence. The ECX bit clears itself after the cancel cancel has been honored. In addition to cancelling the transfer, the ECX treats the cancel as an error condition; thus updating the DMAES register and generating an optional error interrupt. Note: Implementation Specific Parameter.

**Table 4-19. Attribute McIEDMA\_ECX (MclConfigSet) detailed description**

Property	Value
Label	Error Cancel Transfer
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

### 4.5.3 McIEDMAChGroup0Priority (MclConfigSet)

DMA\_CR[GRP0PRI] field configuration. Group 0 priority level when fixed priority group arbitration is enabled. Note: Implementation Specific Parameter.

**Table 4-20. Attribute McIEDMAChGroup0Priority (MclConfigSet) detailed description**

Property	Value
Label	eDMA Channel Group0 Priority
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	0
Invalid	Range <=1 >=0

#### 4.5.4 McIEDMAChGroup1Priority (McIConfigSet)

DMA\_CR[GRP1PRI] field configuration. Group 1 priority level when fixed priority group arbitration is enabled. Note: Implementation Specific Parameter.

**Table 4-21. Attribute McIEDMAChGroup1Priority (McIConfigSet) detailed description**

Property	Value
Label	eDMA Channel Group1 Priority
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	1
Invalid	Range <div>&lt;=1</div> <div>&gt;=0</div>

#### 4.5.5 McIEDMA\_CLM (McIConfigSet)

DMA\_CR[CLM]. Continuous Link Mode. 0 - A minor loop channel link made to itself will go through channel arbitration before being activated again. 1 - A minor loop channel link made to itself will not go through channel arbitration before being activated again. Upon minor loop completion the channel will active again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop. Note: Implementation Specific Parameter.

**Table 4-22. Attribute McIEDMA\_CLM (McIConfigSet) detailed description**

Property	Value
Label	Continuous Link Mode
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

### 4.5.6 McIEDMA\_HALT (MclConfigSet)

DMA\_CR[HALT]. Halt eDMA Operations. 0 - Normal operation. 1 - Stall the start of any new channels. Executing channels are allowed to complete. Channel execution will resume when the HALT bit is cleared. Note: Implementation Specific Parameter.

**Table 4-23. Attribute McIEDMA\_HALT (MclConfigSet) detailed description**

Property	Value
Label	Halt eDMA Operations
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

### 4.5.7 McIEDMA\_HOE (MclConfigSet)

DMA\_CR[HOE]. Halt On Error. 0 - Normal operation. 1 - Any error will cause the HALT bit to be set. Subsequently, all service requests will be ignored until the HALT bit is cleared. Note: Implementation Specific Parameter.

**Table 4-24. Attribute McIEDMA\_HOE (MclConfigSet) detailed description**

Property	Value
Label	Halt On Error
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

### 4.5.8 McIEDMA\_ERGA (MclConfigSet)

DMA\_CR[ERGA]. Enable Round Robin Group Arbitration. 0 - Fixed priority arbitration is used for selection among the groups. 1 - Round robin arbitration is used for selection among the groups. Note: Implementation Specific Parameter.

**Table 4-25. Attribute McIEDMA\_ERGA (MclConfigSet) detailed description**

Property	Value
Label	Round Robin Group Arbitration

*Table continues on the next page...*

**Table 4-25. Attribute McIEDMA\_ERGA (McIConfigSet) detailed description (continued)**

Property	Value
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

### 4.5.9 McIEDMA\_ERCA (McIConfigSet)

DMA\_CR[ERCA]. Enable Round Robin Channel Arbitration. 0 - Fixed-priority arbitration is used for channel selection within each group. 1 - Round-Robin arbitration is used for channel selection within each group. Note: Implementation Specific Parameter.

**Table 4-26. Attribute McIEDMA\_ERCA (McIConfigSet) detailed description**

Property	Value
Label	Round Robin Channel Arbitration
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

### 4.5.10 McIEDMA\_EDBG (McIConfigSet)

DMA\_CR[EDBG]. Enable Debug. 0 - The assertion of the system debug control input is ignored. 1 - The assertion of the system debug control input causes the eDMA to stall the start of a new channel. Executing channels are allowed to complete. Channel execution will resume when either the system debug control input is negated or the EDBG bit is cleared. Note: Implementation Specific Parameter.

**Table 4-27. Attribute McIEDMA\_EDBG (McIConfigSet) detailed description**

Property	Value
Label	Enable Debug
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

## 4.6 Form McIlsrAvailable

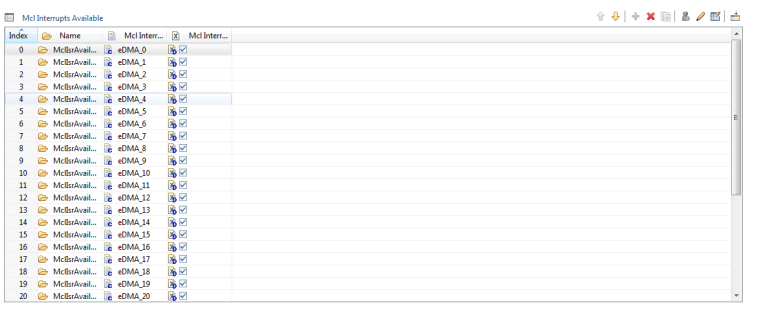


Figure 4-5. Tresos Plugin snapshot for McIlsrAvailable form.

### 4.6.1 McIlsrName (McIlsrAvailable)

Mcl Interrupt Name.

Table 4-28. Attribute McIlsrName (McIlsrAvailable) detailed description

Property	Value
Label	Mcl Interrupt Name
Type	ENUMERATION
Origin	Custom
Symbolic Name	false

### 4.6.2 McIlsrEnabled (McIlsrAvailable)

Switch to indicate if the interrupt is enabled.

- true: Enabled.
- false: Disabled.

Table 4-29. Attribute McIlsrEnabled (McIlsrAvailable) detailed description

Property	Value
Label	Mcl Interrupt Enabled
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	true

## 4.7 Form DMAChannel

All data needed to configure one DMA channel.

Figure 4-6. Tresos Plugin snapshot for DMAChannel form.

### 4.7.1 McIDMAChannelId (DMAChannel)

Id for the current DMA logical Channel. Note: Implementation Specific Parameter.

Table 4-30. Attribute McIDMAChannelId (DMAChannel) detailed description

Property	Value
Label	DMA Channel ID
Type	INTEGER
Origin	Custom
Symbolic Name	false
Invalid	Range <div>&lt;=31</div> <div>&gt;=0</div>

### 4.7.2 DmaHwChannel (DMAChannel)

Select the physical eDMA Channel. NOTE: This is an Implementation Specific Parameter.

**Table 4-31. Attribute DmaHwChannel (DMAChannel) detailed description**

Property	Value
Type	ENUMERATION
Origin	Custom
Symbolic Name	false

### 4.7.3 DMAChannelPriority (DMAChannel)

Priority level for DMA channel. Priorities assigned to channels from the same Group must be unique.

Please read section **Enhanced Direct Memory Access (eDMA)** from the manual for more information

**Table 4-32. Attribute DMAChannelPriority (DMAChannel) detailed description**

Property	Value
Label	DMA Channel Priority
Type	INTEGER
Origin	Custom
Symbolic Name	false
Invalid	Range <=15 >=0

### 4.7.4 ECP (DMAChannel)

Enable channel preemption.

0 (unchecked) - Channel n cannot be suspended by a higher priority channel's service request

1 (checked) - Channel n can be temporarily suspended by a higher priority channel's service request

**Table 4-33. Attribute ECP (DMAChannel) detailed description**

Property	Value
Label	ECP
Type	BOOLEAN
Origin	Custom

*Table continues on the next page...*



**Table 4-33. Attribute ECP (DMAChannel) detailed description (continued)**

Property	Value
Symbolic Name	false
Default	false

### 4.7.5 DPA (DMAChannel)

Disable preemptive ability.

0 (unchecked) - Channel n can suspend a lower priority channel

1 (checked) - Channel n cannot suspend any channel, regardless of the channel's priority.

**Table 4-34. Attribute DPA (DMAChannel) detailed description**

Property	Value
Label	DPA
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

### 4.7.6 EMI (DMAChannel)

Enable Master ID replication.

0 (unchecked) - Master ID replication is disabled

1 (checked) - Master ID replication is enabled

**Table 4-35. Attribute EMI (DMAChannel) detailed description**

Property	Value
Label	EMI
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

### 4.7.7 McIDmaTransferCompletionNotif (DMAChannel)

User callback function NOTE: please use NULL or NULL\_PTR w/o any quotes. If the used string is different from NULL or NULL\_PTR it will be used as the configured function name.

**Table 4-36. Attribute McIDmaTransferCompletionNotif (DMAChannel) detailed description**

Property	Value
Label	Mcl Dma Transfer Completion User Notification
Type	FUNCTION-NAME
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	NULL

### 4.7.8 McIDMAChannelEnable (DMAChannel)

DMA Channel Enable Enables the DMA channel. false - DMA channel is disabled. This mode is primarily used during configuration of the DMA Mux. The DMA has separate channel enables/disables, which should be used to disable or re-configure a DMA channel. true - DMA channel is enabled Note: Implementation Specific Parameter.

**Table 4-37. Attribute McIDMAChannelEnable (DMAChannel) detailed description**

Property	Value
Label	DMA Channel Enable
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

### 4.7.9 McIDMAChannelTriggerEnable (DMAChannel)

DMA Channel Trigger Enable Enables the periodic trigger capability for the triggered DMA channel. false - Triggering is disabled. If triggering is disabled, and the ENBL bit is set, the DMA Channel will simply route the specified source to the DMA channel. (Normal mode) true - Triggering is enabled. If triggering is enabled, and the ENBL bit is set, the DMAMUX is in Periodic Trigger mode. Note: Implementation Specific Parameter.

**Table 4-38. Attribute McIDMAChannelTriggerEnable (DMAChannel) detailed description**

Property	Value
Label	DMA Channel Trigger Enable
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

#### 4.7.10 DmaSource0 (DMAChannel)

Configuration for DMA source slot in DmaMux0 (Physical DMA channels from 0 to 15)

NOTE: This is an Implementation Specific Parameter.

**Table 4-39. Attribute DmaSource0 (DMAChannel) detailed description**

Property	Value
Label	DMA Source 0
Type	ENUMERATION
Origin	Custom
Symbolic Name	false

#### 4.7.11 DmaSource1 (DMAChannel)

Configuration for DMA source slot in DmaMux1 (Physical DMA channels from 16 to 31)

NOTE: This is an Implementation Specific Parameter.

**Table 4-40. Attribute DmaSource1 (DMAChannel) detailed description**

Property	Value
Label	DMA Source 1
Type	ENUMERATION
Origin	Custom
Symbolic Name	false

## 4.8 Form McICrossbarLogicalSlavePorts

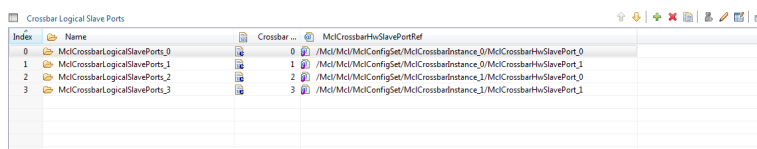


Figure 4-7. Tresos Plugin snapshot for MclCrossbarLogicalSlavePorts form.

### 4.8.1 MclCrossbarLogicalSlavePortId (MclCrossbarLogicalSlavePorts)

Id for the current crossbar logical slave port Note: Implementation Specific Parameter.

**Table 4-41. Attribute MclCrossbarLogicalSlavePortId (MclCrossbarLogicalSlavePorts) detailed description**

Property	Value
Label	Crossbar Logical Slave Port Id
Type	INTEGER
Origin	Custom
Symbolic Name	false
Invalid	Range <div>&lt;=15</div> <div>&gt;=0</div>

### 4.8.2 MclCrossbarHwSlavePortRef (MclCrossbarLogicalSlavePorts)

Select the hardware slave port reference.

**Table 4-42. Attribute MclCrossbarHwSlavePortRef (MclCrossbarLogicalSlavePorts) detailed description**

Property	Value
Type	REFERENCE
Origin	Custom
Enable	true

## 4.9 Form MclCrossbarLogicalMasterPorts

Index	Name	Crossbar ...	MclCrossbarHwSlavePortRef
0	MclCrossbarLogicalSlavePorts_0	0	/Mcl/Mcl/MclConfigSet/MclCrossbarInstance_0/MclCrossbarHwSlavePort_0
1	MclCrossbarLogicalSlavePorts_1	1	/Mcl/Mcl/MclConfigSet/MclCrossbarInstance_0/MclCrossbarHwSlavePort_1
2	MclCrossbarLogicalSlavePorts_2	2	/Mcl/Mcl/MclConfigSet/MclCrossbarInstance_1/MclCrossbarHwSlavePort_0
3	MclCrossbarLogicalSlavePorts_3	3	/Mcl/Mcl/MclConfigSet/MclCrossbarInstance_1/MclCrossbarHwSlavePort_1

Figure 4-8. TresoS Plugin snapshot for MclCrossbarLogicalSlavePorts form.

### 4.9.1 MclCrossbarLogicalSlavePortId (MclCrossbarLogicalSlavePorts)

Id for the current crossbar logical slave port Note: Implementation Specific Parameter.

Table 4-43. Attribute MclCrossbarLogicalMasterPortId (MclCrossbarLogicalMasterPorts) detailed description

Property	Value
Label	Crossbar Logical Master Port Id
Type	INTEGER
Origin	Custom
Symbolic Name	false
Invalid	Range <div>&lt;=15</div> <div>&gt;=0</div>

### 4.9.2 MclCrossbarHwMasterPortRef (MclCrossbarLogicalMasterPorts)

Select the hardware Master port reference.

Table 4-44. Attribute MclCrossbarHwMasterPortRef (MclCrossbarLogicalMasterPorts) detailed description

Property	Value
Type	REFERENCE
Origin	Custom
Enable	true

## 4.10 Form MclCrossbarInstance

Configuration of a crossbar instance.

**Included forms :**

- Form MclCrossbarHwSlavePort

Index	Name	Crossbar ...
0	MciCrossbarInstance_0	AXBS_0
1	MciCrossbarInstance_1	AXBS_1

**Figure 4-9. Tresos Plugin snapshot for MciCrossbarInstance form.**

#### 4.10.1 MclCrossbarHwInstance (MclCrossbarInstance)

Select the physical crossbar instance.

**Table 4-45. Attribute MclCrossbarHwInstance (MclCrossbarInstance) detailed description**

Property	Value
Label	Crossbar Hardware Instance
Type	ENUMERATION
Origin	Custom
Symbolic Name	false

#### 4.10.2 Form MclCrossbarHwSlavePort

### Hardware slave port configuration.

**Is included by form :** [Form MclCrossbarInstance](#)

Index	Name	Slave Port	Priority	Priority	Priority	Priority	Priority	Priority	Priority	Priority	Priority	Lock Conf
1	McICrossbarNoSlavePort	0 UB	0 UB	1 UB	2 UB	3 UB	4 UB	5 UB	6 UB	7 UB		

**Figure 4-10. Tresos Plugin snapshot for MclCrossbarHwSlavePort form.**

#### 4.10.2.1 MclSlavePortNumber (MclCrossbarHwSlavePort)

Slave port number in hardware.

**Table 4-46. Attribute MclSlavePortNumber (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Slave Port Number
Type	INTEGER
Origin	Custom
Symbolic Name	false
Invalid	Range <div>&lt;=7</div> <div>&gt;=0</div>

#### 4.10.2.2 MclCrossbarPrioMaster0 (MclCrossbarHwSlavePort)

Priority of the master 0 on the cross bar.

**Table 4-47. Attribute MclCrossbarPrioMaster0 (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Priority Master0
Type	INTEGER
Origin	Custom
Symbolic Name	false
Invalid	Range <div>&lt;=7</div> <div>&gt;=0</div>

#### 4.10.2.3 MclCrossbarPrioMaster1 (MclCrossbarHwSlavePort)

Priority of the master 1 on the cross bar.

**Table 4-48. Attribute MclCrossbarPrioMaster1 (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Priority Master1
Type	INTEGER
Origin	Custom
Symbolic Name	false
Invalid	Range <div>&lt;=7</div> <div>&gt;=0</div>

#### 4.10.2.4 MclCrossbarPrioMaster2 (MclCrossbarHwSlavePort)

Priority of the master 2 on the cross bar.

**Table 4-49. Attribute MclCrossbarPrioMaster2 (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Priority Master2
Type	INTEGER
Origin	Custom
Symbolic Name	false
Invalid	Range <div> <div>&lt;=7</div> <div>&gt;=0</div> </div>

#### 4.10.2.5 MclCrossbarPrioMaster3 (MclCrossbarHwSlavePort)

Priority of the master 3 on the cross bar.

**Table 4-50. Attribute MclCrossbarPrioMaster3 (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Priority Master3
Type	INTEGER
Origin	Custom
Symbolic Name	false
Invalid	Range <div> <div>&lt;=7</div> <div>&gt;=0</div> </div>

#### 4.10.2.6 MclCrossbarPrioMaster4 (MclCrossbarHwSlavePort)

Priority of the master 4 on the cross bar.

**Table 4-51. Attribute MclCrossbarPrioMaster4 (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Priority Master4

*Table continues on the next page...*



**Table 4-51. Attribute MclCrossbarPrioMaster4 (MclCrossbarHwSlavePort) detailed description (continued)**

Property	Value
Type	INTEGER
Origin	Custom
Symbolic Name	false
Invalid	Range <div> <div>&lt;=7</div> <div>&gt;=0</div> </div>

#### 4.10.2.7 MclCrossbarPrioMaster5 (MclCrossbarHwSlavePort)

Priority of the master 5 on the cross bar.

**Table 4-52. Attribute MclCrossbarPrioMaster5 (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Priority Master5
Type	INTEGER
Origin	Custom
Symbolic Name	false
Invalid	Range <div> <div>&lt;=7</div> <div>&gt;=0</div> </div>

#### 4.10.2.8 MclCrossbarPrioMaster6 (MclCrossbarHwSlavePort)

Priority of the master 6 on the cross bar.

**Table 4-53. Attribute MclCrossbarPrioMaster6 (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Priority Master6
Type	INTEGER
Origin	Custom
Symbolic Name	false
Invalid	Range <div> <div>&lt;=7</div> <div>&gt;=0</div> </div>

#### 4.10.2.9 MclCrossbarPrioMaster7 (MclCrossbarHwSlavePort)

Priority of the master 7 on the cross bar.

**Table 4-54. Attribute MclCrossbarPrioMaster7 (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Priority Master7
Type	INTEGER
Origin	Custom
Symbolic Name	false
Invalid	Range <div>&lt;=7</div> <div>&gt;=0</div>

#### 4.10.2.10 MclCrossbarEnableLock (MclCrossbarHwSlavePort)

Locks the configuration registers for the respective slave port.

**Table 4-55. Attribute MclCrossbarEnableLock (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Lock Configuration Registers
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

#### 4.10.2.11 MclCrossbarHaltLowPrio (MclCrossbarHwSlavePort)

Sets the initial arbitration priority for low power mode requests. Setting this bit will not affect the request for low power mode from attaining highest priority once it has control of the slave ports. Disabled = The low power mode request has the highest priority for arbitration on this slave port; Enabled = The low power mode request has the lowest initial priority for arbitration on this slave port.

**Table 4-56. Attribute MclCrossbarHaltLowPrio (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Halt Low Priority
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

#### 4.10.2.12 MclCrossbarEnablePrioElevM0 (MclCrossbarHwSlavePort)

On this slave port, enable priority elevation for master 0. If enabled, the master is able to elevate its priority to the highest.

**Table 4-57. Attribute MclCrossbarEnablePrioElevM0 (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Enable Priority elevation Master 0
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

#### 4.10.2.13 MclCrossbarEnablePrioElevM1 (MclCrossbarHwSlavePort)

LocksOn this slave port, enable priority elevation for master 1. If enabled, the master is able to elevate its priority to the highest.

**Table 4-58. Attribute MclCrossbarEnablePrioElevM1 (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Enable Priority elevation Master 1
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

#### 4.10.2.14 MclCrossbarEnablePrioElevM2 (MclCrossbarHwSlavePort)

LocksOn this slave port, enable priority elevation for master 2. If enabled, the master is able to elevate its priority to the highest.

**Table 4-59. Attribute MclCrossbarEnablePrioElevM2 (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Enable Priority elevation Master 2
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

#### 4.10.2.15 MclCrossbarEnablePrioElevM3 (MclCrossbarHwSlavePort)

LocksOn this slave port, enable priority elevation for master 3. If enabled, the master is able to elevate its priority to the highest.

**Table 4-60. Attribute MclCrossbarEnablePrioElevM3 (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Enable Priority elevation Master 3
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

#### 4.10.2.16 MclCrossbarEnablePrioElevM4 (MclCrossbarHwSlavePort)

LocksOn this slave port, enable priority elevation for master 4. If enabled, the master is able to elevate its priority to the highest.

**Table 4-61. Attribute MclCrossbarEnablePrioElevM4 (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Enable Priority elevation Master 4
Type	BOOLEAN

*Table continues on the next page...*

**Table 4-61. Attribute MclCrossbarEnablePrioElevM4 (MclCrossbarHwSlavePort) detailed description (continued)**

Property	Value
Origin	Custom
Symbolic Name	false
Default	false

#### 4.10.2.17 MclCrossbarEnablePrioElevM5 (MclCrossbarHwSlavePort)

LocksOn this slave port, enable priority elevation for master 5. If enabled, the master is able to elevate its priority to the highest.

**Table 4-62. Attribute MclCrossbarEnablePrioElevM5 (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Enable Priority elevation Master 5
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

#### 4.10.2.18 MclCrossbarEnablePrioElevM6 (MclCrossbarHwSlavePort)

LocksOn this slave port, enable priority elevation for master 6. If enabled, the master is able to elevate its priority to the highest.

**Table 4-63. Attribute MclCrossbarEnablePrioElevM6 (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Enable Priority elevation Master 6
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

#### 4.10.2.19 MclCrossbarEnablePrioElevM7 (MclCrossbarHwSlavePort)

LocksOn this slave port, enable priority elevation for master 7. If enabled, the master is able to elevate its priority to the highest.

**Table 4-64. Attribute MclCrossbarEnablePrioElevM7 (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Enable Priority elevation Master 7
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

#### 4.10.2.20 MclCrossbarEnableFixedPrio (MclCrossbarHwSlavePort)

On this slave port, select the arbitraion mode: if enabled the arbitration mode will be FIXED PRIORITY, if disabled it will be ROUND ROBIN.

**Table 4-65. Attribute MclCrossbarEnableFixedPrio (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Enable Fixed Priority Arbitration
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

#### 4.10.2.21 MclCrossbarParkingControl (MclCrossbarHwSlavePort)

Determines the slave port's parking control. The low-power park feature results in an overall power savings if the slave port is not saturated; however, this forces an extra latency clock when any master tries to access the slave port while not in use because it is not parked on any master.

**Table 4-66. Attribute MclCrossbarParkingControl (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Parking Control
Type	ENUMERATION
Origin	Custom
Symbolic Name	false

#### 4.10.2.22 MclCrossbarParkField (MclCrossbarHwSlavePort)

Determines which master port the current slave port parks on when no masters are actively making requests and the MclCrossbarParkingControl=ParkField.

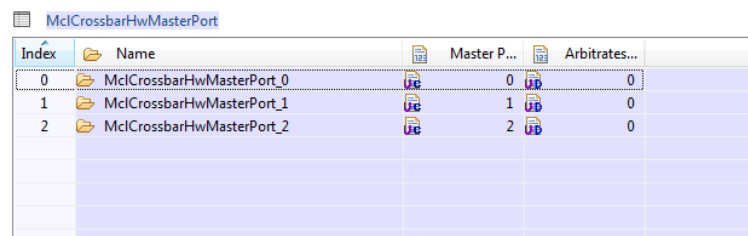
**Table 4-67. Attribute MclCrossbarParkField (MclCrossbarHwSlavePort) detailed description**

Property	Value
Label	Park field
Type	ENUMERATION
Origin	Custom
Symbolic Name	false

### 4.10.3 Form MclCrossbarHwMasterPort

Hardware Master port configuration.

Is included by form : [Form MclCrossbarInstance](#)



Index	Name	Master P...	Arbitrates...
0	MclCrossbarHwMasterPort_0	0	0
1	MclCrossbarHwMasterPort_1	1	0
2	MclCrossbarHwMasterPort_2	2	0

**Figure 4-11. Tresos Plugin snapshot for MclCrossbarHwMasterPort form.**

#### 4.10.3.1 MclMasterPortNumber (MclCrossbarHwMasterPort)

Master port number in hardware.

**Table 4-68. Attribute MclMasterPortNumber (MclCrossbarHwMasterPort) detailed description**

Property	Value
Label	Master Port Number
Type	INTEGER
Origin	Custom
Symbolic Name	false
Invalid	Range <div>&lt;=7</div> <div>&gt;=0</div>

#### 4.10.3.2 MclCrossbarArbitrates (MclCrossbarHwMasterPort)

Priority of the master 0 on the cross bar.

**Table 4-69. Attribute MclCrossbarArbitrates (MclCrossbarHwMasterPort) detailed description**

Property	Value
Label	Arbitrates On Undefined Length Bursts
Type	INTEGER
Origin	Custom
Symbolic Name	false
Invalid	Range <div>&lt;=4</div> <div>&gt;=0</div>

## 4.11 Form CommonPublishedInformation

Common container, aggregated by all modules. It contains published information about vendor and versions.



CommonPublishedInformation	
Name	Value
ArReleaseMajorVersion	4
ArReleaseMinorVersion	2
ArReleaseRevisionVersion	2
ModuleId	255
SwMajorVersion	1
SwMinorVersion	0
SwPatchVersion	0
VendorApiInfix	
VendorId	43

**Figure 4-12. Tresa Plugin snapshot for CommonPublishedInformation form.**

### 4.11.1 ArReleaseMajorVersion (CommonPublishedInformation)

Major version number of AUTOSAR specification on which the appropriate implementation is based on.

**Table 4-70. Attribute ArReleaseMajorVersion (CommonPublishedInformation) detailed description**

Property	Value
Label	AUTOSAR Major Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	4
Invalid	Range <div>&gt;=4</div> <div>&lt;=4</div>

### 4.11.2 ArReleaseMinorVersion (CommonPublishedInformation)

Minor version number of AUTOSAR specification on which the appropriate implementation is based on.

**Table 4-71. Attribute ArReleaseMinorVersion (CommonPublishedInformation) detailed description**

Property	Value
Label	AUTOSAR Minor Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	2
Invalid	Range >=2 <=2

### 4.11.3 ArReleaseRevisionVersion (CommonPublishedInformation)

Revision version number of AUTOSAR specification on which the appropriate implementation is based on.

**Table 4-72. Attribute ArReleaseRevisionVersion (CommonPublishedInformation) detailed description**

Property	Value
Label	AUTOSAR Release Revision Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	2
Invalid	Range >=2 <=2

### 4.11.4 ModuleId (CommonPublishedInformation)

Module ID of this module from Module List.

**Table 4-73. Attribute ModuleId (CommonPublishedInformation) detailed description**

Property	Value
Label	Module Id
Type	INTEGER_LABEL

*Table continues on the next page...*

**Table 4-73. Attribute ModuleId (CommonPublishedInformation) detailed description (continued)**

Property	Value
Origin	Custom
Symbolic Name	false
Default	255
Invalid	Range >=255 <=255

### 4.11.5 SwMajorVersion (CommonPublishedInformation)

Major version number of the vendor specific implementation of the module. The numbering is vendor specific.

**Table 4-74. Attribute SwMajorVersion (CommonPublishedInformation) detailed description**

Property	Value
Label	Software Major Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	1
Invalid	Range >=1 <=1

### 4.11.6 SwMinorVersion (CommonPublishedInformation)

Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.

**Table 4-75. Attribute SwMinorVersion (CommonPublishedInformation) detailed description**

Property	Value
Label	Software Minor Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	0

*Table continues on the next page...*

**Table 4-75. Attribute SwMinorVersion (CommonPublishedInformation) detailed description (continued)**

Property	Value
Invalid	Range $\geq 0$ $\leq 0$

### 4.11.7 SwPatchVersion (CommonPublishedInformation)

Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

**Table 4-76. Attribute SwPatchVersion (CommonPublishedInformation) detailed description**

Property	Value
Label	Software Patch Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	0
Invalid	Range $\geq 0$ $\leq 0$

### 4.11.8 VendorApiInfix (CommonPublishedInformation)

In driver modules which can be instantiated several times on a single ECU, BSW00347 requires that the name of APIs is extended by the VendorId and a vendor specific name. This parameter is used to specify the vendor specific name. In total, the implementation specific name is generated as follows:

<ModuleName>\_>VendorId>\_<VendorApiInfix><Api name from SWS>. E.g. assuming that the VendorId of the implementor is 123 and the implementer chose a VendorApiInfix of "v11r456" a api name Can\_Write defined in the SWS will translate to Can\_123\_v11r456Write. This parameter is mandatory for all modules with upper multiplicity > 1. It shall not be used for modules with upper multiplicity =1.

**Table 4-77. Attribute VendorApiInfix (CommonPublishedInformation) detailed description**

Property	Value
Label	Vendor Api Infix

*Table continues on the next page...*

**Table 4-77. Attribute VendorApilnfix (CommonPublishedInformation) detailed description (continued)**

Property	Value
Type	STRING_LABEL
Origin	Custom
Symbolic Name	false
Default	
Enable	false

### 4.11.9 VendorId (CommonPublishedInformation)

Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list.

**Table 4-78. Attribute VendorId (CommonPublishedInformation) detailed description**

Property	Value
Label	Vendor Id
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	43
Invalid	Range >=43 <=43



**How to Reach Us:****Home Page:**[nxp.com](http://nxp.com)**Web Support:**[nxp.com/support](http://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](http://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  $\mu$ Vision 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.

© 2016 NXP B.V.

Document Number UM47MCLASR4.2 Rev002R1.0.0  
Revision 1.2