User Manual

for S32K14X MCL Driver

Document Number: UM2MCLASR4.3 Rev0001R1.0.1
Rev. 1.0

Contents

Section number **Title** **Page**

Chapter 1
Revision History

Chapter 2
Introduction

Chapter 3
Driver

Chapter 4
Tresos Configuration Plug-in

Chapter 1
Revision History

Table 1-1. Revision History

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

Chapter 2
Introduction

This User Manual describes NXP Semiconductors AUTOSAR MicroController Library (MCL) for S32K14X .

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

2.1 Supported Derivatives

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

Table 2-1. S32K14X Derivatives

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

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

2.2 Overview

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

AUTOSAR

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

2.3 About this Manual

This Technical Reference employs the following typographical conventions:

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

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

Notes and warnings are shown as below:

Note

This is a note.

2.4 Acronyms and Definitions

Table 2-2. Acronyms and Definitions

Term	Definition
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
ASM	Assembler
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

Table continues on the next page...

Table 2-2. Acronyms and Definitions (continued)

Term	Definition
N/A	Not Applicable
MCL	Micro Controller Library
FTM	FlexTimer Module

2.5 Reference List

Table 2-3. Reference List

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

Chapter 3
Driver

3.1 Requirements

MCL is a complex driver, so there are no AUTOSAR requirements regarding this module. For the S32K14X platform, the MCL module configures the DMA and DMAMUX functionalities.

3.2 Driver Design Summary

The MCL driver configures the direct memory access and direct memory access multiplexer. Also, MCL it is a container for common functionalities: FTM, LPIT, LPTMR common files are included in MCL because they are used by several modules(eg. ICU, PWM). If some modules need the FTM, LPIT, LPTMR common files and they do not need the DMA or LMEM functionality, they should only include the FTM, LPIT, LPTMR common files, whithout 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 S32K14X family

Hardware IP	Description
eDMA	Enhanced Direct Memory Access. The platform includes 1 DMA instance having 16 hardware channels.
DMAMUX	Direct Memory Access Multiplexer. The platform includes 1 DMAMUX instances, having 16 hardware channels.
FTM	FlexTimer Module. MCL includes the shared lower level functionalities for FTM
LPIT	Low Power Interrupt Timer. MCL includes the shared lower level functionalities for LPIT
LPTMR	Low Power Timer. MCL includes the shared lower level functionalities for LPTMR
Lmem	local memory controller. MCL includes the shared lower level functionalities for Lmem
FlexIO	Flexible I/O. MCL includes the shared lower level functionalities for FlexIO

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.

- Mcl driver can not return from interrupt error handling when eDma_ES appears error priority.
- Mcl driver will be called all higher lever function if appears corresponding error.

3.6 Driver Usage and Configuration tips

MCL feature selecting

The MCL module is a non-AUTOSAR driver providing support for several hardware features.The user should configure only the hardware feature that is needed in his project. For example: in case the DMA support is needed, configure EnableDMA as true.

If no MCL hardware feature is used(DMA, AXBS, TRGMUX, LMEM), the MCL module is stil needed in the SMCAL architecture as MCL is a library containing the timer shared files used by other SMCAL modules(GPT, ICU, PWM). In this case it's not necessary to configure MCL and it's not necessairy to call Mcl_Init because the timer hardware is configured and initialized by other SMCAL modules.

MCL DMA allignment

All source reads and destination writes must be configured to the natural boundary of the programmed transfer size.

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.

MCL DMA major loop and minor loop transfers

The figure bellow depicts the major and minor loop DMA transfer related to a DMA request.

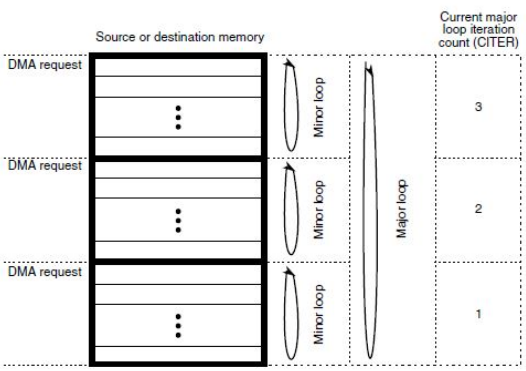

Figure 3-1. DMA major and minor loop

The figure bellow depicts the major and minor loop DMA transfer related to TCD configuration.

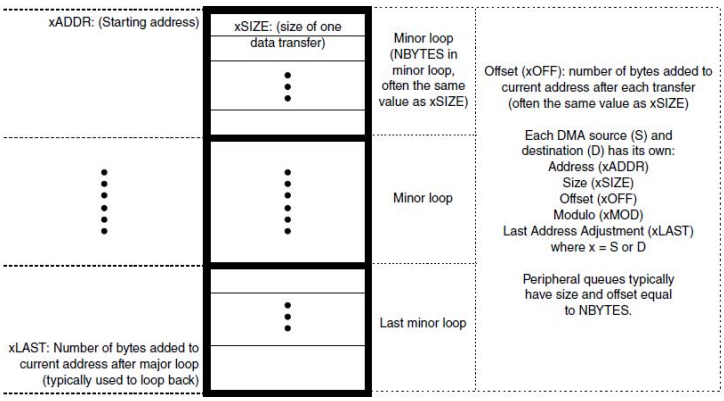

Figure 3-2. DMA major and minor loop

MCL DMA memory to memory simple transfer

The recommendations and examples bellow refer to the use case when a simple memory to memory transfer is needed.

Configuration settings needed in Tresos:

- Configure *EnableDma* as True, configure at least one DmaInstance and one DmaChannel correspondig to the configured DmaInstance
- Configure *MclDmaTransferCompletionNotif* in case a channel transfer notification is needed at half transfer or the end of the transfer
- Configure the transfer completion ISR corresponding to the used hardware channel as enabled in the *MclIsrEnabled* container
- Set *MclDMAChannelEnable* as true, configure the DmaSource corresponding to the needed hardware channel as ALWAYS_ENABLED

Runtime steps for a DMA transfer:

- Prepare the transfer configuration pointer

  Example:

  *tcd_config_ptr->u32saddr = src_buffer1 /*source */*

  tcd_config_ptr->u32daddr = dest_buffer1 / destination */*

  tcd_config_ptr->u32ssize = DMA_SIZE_1BYTE

  tcd_config_ptr->u32dsize = DMA_SIZE_1BYTE

  tcd_config_ptr->u32soff = DMA_OFFSET_8_BITS

  tcd_config_ptr->u32doff = DMA_OFFSET_8_BITS

tcd_config_ptr->u32smod = DISABLE_FEATURE

tcd_config_ptr->u32dmod = DISABLE_FEATURE

tcd_config_ptr->u32num_bytes = BYTES_TO_BE_TRANSFERRED

tcd_config_ptr->u32iter = 1 / transfer major loop count - TCD BITER */*

- Use the API *Mcl_DmaConfigChannel* to configure in hardware the transfer descriptor information

  Example: Mcl_DmaConfigChannel(MCL_DMA_LOGICAL_CHANNEL_0,tcd_config_ptr)

- Use the API *Mcl_DmaEnableNotification* to enable the callback notification at half transfer or at the end of the transfer

  Example: Mcl_DmaEnableNotification(MCL_DMA_LOGICAL_CHANNEL_0, MCL_DMA_TRANSFER_COMPLETE)

- Use the API *Mcl_DmaStartChannel* to start the transfer by software

  Example: Mcl_DmaStartChannel(MCL_DMA_LOGICAL_CHANNEL_0)

- End of the transfer will be notified by the notification callback if configured and enabled or the user can get the transfer status by polling it with Mcl_DmaIsTransferCompleted.

MCL DMA memory to peripheral transfer

The recommendations and examples bellow refer to the use case when a memory to peripheral (such as SPI) transfer is needed.

Configuration settings needed in Tresos:

- Configure *EnableDma* as True, configure at least one DmaInstance and one DmaChannel correspondig to the configured DmaInstance
- Configure *MclDmaTransferCompletionNotif* in case a channel transfer notification is needed at half transfer or the end of the transfer
- Configure the transfer completion ISR corresponding to the used hardware channel as enabled in the *MclIsrEnabled* container
- Set *MclDMAChannelEnable* as true, configure the DmaSource corresponding to the hardware channel as the peripheral needed

Runtime steps for a DMA transfer:

- Prepare the transfer configuration pointer

  Example:

*tcd_config_ptr->u32saddr = src_buffer1 /*source */*

tcd_config_ptr->u32daddr = dest_buffer1 / destination */*

tcd_config_ptr->u32ssize = DMA_SIZE_1BYTE

tcd_config_ptr->u32dsize = DMA_SIZE_1BYTE

tcd_config_ptr->u32soff = DMA_OFFSET_8_BITS

tcd_config_ptr->u32doff = DMA_OFFSET_8_BITS

tcd_config_ptr->u32smod = DISABLE_FEATURE

tcd_config_ptr->u32dmod = DISABLE_FEATURE

tcd_config_ptr->u32num_bytes = BYTES_TO_BE_TRANSFERRED

tcd_config_ptr->u32iter = 1 / transfer major loop count - TCD BITER */*

- Use the API *Mcl_DmaConfigChannel* to configure in hardware the transfer descriptor information

  Example: Mcl_DmaConfigChannel(MCL_DMA_LOGICAL_CHANNEL_0,tcd_config_ptr)

- Use the API *Mcl_DmaEnableNotification* to enable the callback notification at half transfer or at the end of the transfer

  Example: Mcl_DmaEnableNotification(MCL_DMA_LOGICAL_CHANNEL_0, MCL_DMA_TRANSFER_COMPLETE)

- Use the API *Mcl_DmaEnableHwRequest* to enable the peripheral request. If this API is not used, the edge from the peripheral will be ignored and the transfer will never be started.

  Example: Mcl_DmaEnableHwRequest(MCL_DMA_LOGICAL_CHANNEL_0)

- The transfer will be started by the first edge from the hardware peripheral
- End of the transfer will be notified by the notification callback if configured and enabled or the user can get the transfer status by polling it with Mcl_DmaIsTransferCompleted.

MCL DMA memory to memory transfer with channel linking

The recommendations and examples bellow refer to the use case when two memory to memory transfers are needed, each transfer is executed by a MCL channel and the channel linking feature is used. The channel linking feature enables the user to program two transfers using two different channels and the second transfer being actived by hardware when the first transfer in completed.

Configuration settings needed in Tresos:

- Configure *EnableDma* as True, configure at least one DmaInstance and two DmaChannels correspondig to the configured DmaInstance
- Configure *MclDmaTransferCompletionNotif* in case a channel transfer notification is needed at half transfer or the end of the transfer
- Configure the transfer completion ISR corresponding to the used hardware channels as enabled in the *MclIsrEnabled* container
- Set *MclDMAChannelEnable* as true, configure the DmaSource corresponding to the needed hardware channels as ALWAYS_ENABLED

Runtime steps for a DMA transfer:

- Prepare the transfer configuration pointer 1

  Example:

  *tcd_config_ptr1->u32saddr = src_buffer1 /*source */*

  tcd_config_ptr1->u32daddr = dest_buffer1 / destination */*

  tcd_config_ptr1->u32ssize = DMA_SIZE_1BYTE

  tcd_config_ptr1->u32dsize = DMA_SIZE_1BYTE

  tcd_config_ptr1->u32soff = DMA_OFFSET_8_BITS

  tcd_config_ptr1->u32doff = DMA_OFFSET_8_BITS

  tcd_config_ptr1->u32smod = DISABLE_FEATURE

  tcd_config_ptr1->u32dmod = DISABLE_FEATURE

  tcd_config_ptr1->u32num_bytes = BYTES_TO_BE_TRANSFERRED

  tcd_config_ptr1->u32iter = 1 / transfer major loop count - TCD BITER */*

- Prepare the transfer configuration pointer 2

  Example:

  *tcd_config_ptr2->u32saddr = src_buffer2 /*source */*

  tcd_config_ptr2->u32daddr = dest_buffer1+BYTES_TO_BE_TRANSFERRED/ destination */*

  tcd_config_ptr2->u32ssize = DMA_SIZE_1BYTE

  tcd_config_ptr2->u32dsize = DMA_SIZE_1BYTE

  tcd_config_ptr2->u32soff = DMA_OFFSET_8_BITS

tcd_config_ptr2->u32doff = DMA_OFFSET_8_BITS

tcd_config_ptr2->u32smod = DISABLE_FEATURE

tcd_config_ptr2->u32dmod = DISABLE_FEATURE

tcd_config_ptr2->u32num_bytes = BYTES_TO_BE_TRANSFERRED

tcd_config_ptr2->u32iter = 1 / transfer major loop count - TCD BITER */*

- Use the API *Mcl_DmaConfigLinkedChannel* to configure in hardware the transfer descriptor information using the channel linking feature for CHANNEL_0

  Example: Mcl_DmaConfigLinkedChannel(MCL_DMA_LOGICAL_CHANNEL_0,tcd_config_ptr1,MCL_DMA_LOGICAL_CHANNEL_1)

- Use the API *Mcl_DmaConfigChannel* to configure in hardware the transfer descriptor information for CHANNEL_1

  Example: Mcl_DmaConfigChannel(MCL_DMA_LOGICAL_CHANNEL_1, tcd_config_ptr2)

- Use the API *Mcl_DmaEnableNotification* to enable the callback notification at half transfer or at the end of the transfer

  Example: Mcl_DmaEnableNotification(MCL_DMA_LOGICAL_CHANNEL_0, MCL_DMA_TRANSFER_COMPLETE)

  Mcl_DmaEnableNotification(MCL_DMA_LOGICAL_CHANNEL_1, MCL_DMA_TRANSFER_COMPLETE)

- Use the API *Mcl_DmaStartChannel* to start the transfer by software

  Example: Mcl_DmaStartChannel(MCL_DMA_LOGICAL_CHANNEL_0)

- End of the transfer will be notified by the notification callback if configured and enabled or the user can get the transfer status by polling it with Mcl_DmaIsTransferCompleted.
- When the transfer for CHANNEL_0 is finished, then the hardware will automatically activate the transfer for CHANNEL_1

MCL DMA memory to memory transfer with scatter gatther

The recommendations and examples bellow refer to the use case: a memory to memory transfer is needed with the scatter gather feature. The scatter gather feature enables the user to program different transfers using the same MCL channel. When the channel has finished the first transfer, if the scatter gather feature is used, the channel transfer

descriptor (TCD) will be reconfigured with the values from a fixed address, thus enabling a second transfer on the same channel. For example, the feature enables a MCL channel to scatter the DMA data to multiple destinations or gather it from multiple sources.

Configuration settings needed in Tresos:

- Configure *EnableDma* as True, configure at least one DmaInstance and 1 DmaChannel correspondig to the configured DmaInstance
- Configure *MclDmaTransferCompletionNotif* in case a channel transfer notification is needed at half transfer or the end of the transfer
- Configure the transfer completion ISR corresponding to the used hardware channel as enabled in the *MclIsrEnabled* container
- Set *MclDMAChannelEnable* as true, configure the DmaSource corresponding to the needed hardware channel as ALWAYS_ENABLED

Runtime steps for a DMA transfer:

- Prepare the transfer configuration pointer 1

  Example:

  *tcd_config_ptr1->u32saddr = src_buffer1 /*source */*

  tcd_config_ptr1->u32daddr = dest_buffer1 / destination */*

  tcd_config_ptr1->u32ssize = DMA_SIZE_1BYTE

  tcd_config_ptr1->u32dsize = DMA_SIZE_1BYTE

  tcd_config_ptr1->u32soff = DMA_OFFSET_8_BITS

  tcd_config_ptr1->u32doff = DMA_OFFSET_8_BITS

  tcd_config_ptr1->u32smod = DISABLE_FEATURE

  tcd_config_ptr1->u32dmod = DISABLE_FEATURE

  tcd_config_ptr1->u32num_bytes = BYTES_TO_BE_TRANSFERRED

  tcd_config_ptr1->u32iter = 1 / transfer major loop count - TCD BITER */*

- Prepare the transfer configuration pointer 2

  Example:

  *tcd_config_ptr2->u32saddr = src_buffer2 /*source */*

  tcd_config_ptr2->u32daddr = dest_buffer1+BYTES_TO_BE_TRANSFERRED/ destination */*

  tcd_config_ptr2->u32ssize = DMA_SIZE_1BYTE

tcd_config_ptr2->u32dsize = DMA_SIZE_1BYTE

tcd_config_ptr2->u32soff = DMA_OFFSET_8_BITS

tcd_config_ptr2->u32doff = DMA_OFFSET_8_BITS

tcd_config_ptr2->u32smod = DISABLE_FEATURE

tcd_config_ptr2->u32dmod = DISABLE_FEATURE

tcd_config_ptr2->u32num_bytes = BYTES_TO_BE_TRANSFERRED

tcd_config_ptr2->u32iter = 1 / transfer major loop count - TCD BITER */*

- Allocate a RAM buffer to be used for the RAM descriptor which will be reloaded into the channel hardware TCD after the first channel transfer is finished The buffer should be aligned to 32 bytes and have the same size as a hardware TCD.

  Example: VAR_ALIGN(VAR(uint32, AUTOMATIC) tcd_memory[8], 32)

- Set a pointer to the allocated RAM area.

  Example: pNextTcd = tcd_memory

- Use the API *Mcl_DmaConfigTcd* to prepare the RAM TCD which will be reloaded into the channel hardware TCD after the first channel transfer is finished.

  Example: Mcl_DmaConfigTcd(pNextTcd, tcd_config_ptr2)

- Use the API *Mcl_DmaTcdSetFlags* to set the START bit in the RAM TCD

  Example: Mcl_DmaTcdSetFlags(pNextTcd, DMA_TCD_START_U32)

- Use the API *Mcl_DmaConfigScatterGatherChannel* to configure the transfer descriptor and enable/configure the scatter gather feature.

  Example: Mcl_DmaConfigScatterGatherChannel(MCL_DMA_LOGICAL_CHANNEL_0, tcd_config_ptr1, pNextTcd)

- Use the API *Mcl_DmaEnableNotification* to enable the callback notification at half transfer or at the end of the transfer

  Example: Mcl_DmaEnableNotification(MCL_DMA_LOGICAL_CHANNEL_0, MCL_DMA_TRANSFER_COMPLETE)

  Mcl_DmaEnableNotification(MCL_DMA_LOGICAL_CHANNEL_1, MCL_DMA_TRANSFER_COMPLETE)

- Use the API *Mcl_DmaStartChannel* to start the transfer by software

Example: Mcl_DmaStartChannel(MCL_DMA_LOGICAL_CHANNEL_0)

- End of the transfer will be notified by the notification callback if configured and enabled or the user can get the transfer status by polling it with Mcl_DmaIsTransferCompleted.

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: • 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.
Mcl_DmaGetChannelErrorStatus	MCL_DMA_E_DESCRIPTOR	Description of the error condition in the hardware: This error is reported when the DMA TCD is

Table continues on the next page...

Table 3-2. Runtime Errors (continued)

Function	Error Code	Condition triggering the error
		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.
Mcl_DmaGetGlobalErrorStatus	MCL_DMA_E_ECC	The error is reported when the UCE bit of DMA_ES is set becasue 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 becasue of an uncorrectable ECC error during channel execution.
Mcl_DmaGetGlobalErrorStatus	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.

Table continues on the next page...

Table 3-2. Runtime Errors (continued)

Function	Error Code	Condition triggering the 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 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 continues on the next page...

Table 3-2. Runtime Errors (continued)

Function	Error Code	Condition triggering the error
Mcl_DmaGetChannelErrorStatus	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 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.
Mcl_DmaGetGlobalErrorStatus	MCL_DMA_E_UNRECOGNIZED	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.
Mcl_DmaGetChannelErrorStatus	MCL_DMA_E_UNRECOGNIZED	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.

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

Name	MCL_ACKNOWLEDGEINTERRUPT_ID_U8
Initializer	(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

Name	MCL_AR_RELEASE_MAJOR_VERSION
Initializer	4

3.8.1.3 Define MCL_AR_RELEASE_MINOR_VERSION

Implements: Mcl_interface

Table 3-5. Define MCL_AR_RELEASE_MINOR_VERSION Description

Name	MCL_AR_RELEASE_MINOR_VERSION
Initializer	3

3.8.1.4 Define MCL_AR_RELEASE_REVISION_VERSION

Implements: Mcl_interface

Table 3-6. Define MCL_AR_RELEASE_REVISION_VERSION Description

Name	MCL_AR_RELEASE_REVISION_VERSION
Initializer	1

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

Name	MCL_CONFIG_CH_ID_U8
Initializer	(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

Name	MCL_CONFIG_LINK_CH_ID_U8
Initializer	(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

Name	MCL_CONFIG_LINK_TCD_ID_U8
Initializer	(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

Name	MCL_CONFIG_SCA_CH_ID_U8
Initializer	(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

Name	MCL_CONFIG_SCA_LINK_CH_ID_U8
Initializer	(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

Name	MCL_CONFIG_SCA_LINK_TCD_ID_U8
Initializer	(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

Name	MCL_CONFIG_SCA_TCD_ID_U8
Initializer	(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

Name	MCL_CONFIG_TCD_ID_U8
Initializer	(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

Name	MCL_DISABLENOTIFICATION_ID_U8
Initializer	(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

Name	MCL_DMACLEARDONE_ID_U8
Initializer	(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

Name	MCL_DMAGETCHANNELTCDADDRESS_ID_U8
Initializer	(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

Name	MCL_DMAGETINTERRUPTREQUEST_ID_U8
Initializer	(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

Name	MCL_DMAGETPHYSICALCHANNEL_ID_U8
Initializer	(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

Name	MCL_DMATCDGETINTMAJ_ID_U8
Initializer	(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

Name	MCL_E_ALREADY_INITIALIZED_U8
Initializer	(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

Name	MCL_E_INVALID_CHANNEL_U8
Initializer	(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

Name	MCL_E_PARAM_CONFIG_U8
Initializer	(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

Name	MCL_E_PARAM_NOTIFICATION_NULL_U8
Initializer	(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

Name	MCL_E_PARAM_POINTER_U8
Initializer	(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

Name	MCL_E_PARAM_VINFO_U8
Initializer	(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

Name	MCL_E_UNEXPECTED_ISR_U8
Initializer	(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

Name	MCL_E_UNINIT_U8
Initializer	(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

Name	MCL_ENABLENOTIFICATION_ID_U8
Initializer	(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

Name	MCL_GETVERSIONINFO_ID_U8
Initializer	(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

Name	MCL_GET_GLOBAL_ERR_STATUS_ID_U8
Initializer	(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

Name	MCL_GET_CH_ERR_STATUS_ID_U8
Initializer	(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

Name	MCL_DMA_NO_CHANNEL_U16
Initializer	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

Name	MCL_DMA_CHANNEL_NOT_CONFIGURED_U8
Initializer	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

Name	MCL_INIT_ID_U8
Initializer	(uint8)0x23U

3.8.1.34 Define MCL_MODULE_ID

Implements: Mcl_interface

Table 3-36. Define MCL_MODULE_ID Description

Name	MCL_MODULE_ID
Initializer	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

Name	MCL_SET_PRI_ID_U8
Initializer	(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

Name	MCL_START_CH_ID_U8
Initializer	(uint8)0x2BU

3.8.1.37 Define MCL_SW_MAJOR_VERSION

Implements: Mcl_interface

Table 3-39. Define MCL_SW_MAJOR_VERSION Description

Name	MCL_SW_MAJOR_VERSION
Initializer	1

3.8.1.38 Define MCL_SW_MINOR_VERSION

Implements: Mcl_interface

Table 3-40. Define MCL_SW_MINOR_VERSION Description

Name	MCL_SW_MINOR_VERSION
Initializer	0

3.8.1.39 Define MCL_SW_PATCH_VERSION

Implements: Mcl_interface

Table 3-41. Define MCL_SW_PATCH_VERSION Description

Name	MCL_SW_PATCH_VERSION
Initializer	1

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

Name	MCL_TRANSF_ACTIVE_ID_U8
Initializer	(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

Name	MCL_TRANSF_COMPL_ID_U8
Initializer	(uint8)0x2CU

3.8.1.42 Define MCL_VENDOR_ID

Implements: Mcl_interface

Table 3-44. Define MCL_VENDOR_ID Description

Name	MCL_VENDOR_ID
Initializer	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

Name	DMA_TCD_DONE_U8
Initializer	((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

Name	DMA_TCD_ACTIVE_U8
Initializer	((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

Name	DMA_TCD_MAJOR_E_LINK_U8
Initializer	((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

Name	DMA_TCD_E_SG_U8
Initializer	((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

Name	DMA_TCD_DISABLE_REQ_U8
Initializer	((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

Name	DMA_TCD_INT_HALF_U8
Initializer	((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

Name	DMA_TCD_INT_MAJOR_U8

Table continues on the next page...

Table 3-51. Define DMA_TCD_INT_MAJOR_U8 Description (continued)

Initializer	((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

Name	DMA_TCD_START_U8
Initializer	((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

Name	MCL_GET_CRT_ITER_CH_ID_U8
Initializer	(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

Name	MCL_GET_STRT_ITER_CH_ID_U8
Initializer	(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

Name	MCL_UPDATE_DEST_CH_ID_U8
Initializer	(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

Name	MCL_UPDATE_ITER_ID_U8
Initializer	(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	1	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_16BYTES	4	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.

Table continues on the next page...

Table 3-60. Enumeration Mcl_DmaChannelErrorStatusType Values (continued)

Name	Initializer	Description
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 MclDevErorDetect 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.
const`Mcl_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.
const`Mcl_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.
const`Mcl_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.
const`Mcl_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.
const`Mcl_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.
const`Mcl_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.
const`Mcl_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.
const`Mcl_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_DmaDisableeNotification_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 adress for the corresponding tcd.

Return: The adress 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 phyisical 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_DmaIsTransferActive

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_DmaIsTransferActive Arguments

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

3.8.3.18 Function Mcl_DmaIsTransferCompleted

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.
uint32	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 geting 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. If the hardware allows for only one usage of the register, the driver module implementing that functionality is responsible for initializing the register. The initialization function of this module shall always have a pointer as a parameter, even though for Variant PC no configuration set shall be given. Instead 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
const`Mcl_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.
`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 phyisical 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 .

3.8.3.51 Function Mcl_TrgMuxConfigInput

This function configure a trigger in a TRGMUX register.

Details:

This function is used for configuring a trigger in a TRGMUX register

Return: void.

Prototype: `FUNC(void, MCL_CODE) Mcl_TrgMuxConfigInput(VAR(Mcl_TrgMuxRegisterIndexType, AUTOMATIC) registerIndex, VAR(Mcl_TrgMuxSelectionNrType, AUTOMATIC) selNumber, VAR(Mcl_TrgMuxTriggerType, AUTOMATIC) trigger)`

Table 3-111. Mcl_TrgMuxConfigInput Arguments

Type	Name	Direction	Description
`Mcl_TrgMuxRegisterIndexType`	registerIndex	**input**	TRGMUX register index.
`Mcl_TrgMuxSelectionNrType`	selNumber	**input**	number of the input configured(sel0,sel1,sel2,sel3).
`Mcl_TrgMuxTriggerType`	trigger	**input**	trigger to be configured.

3.8.3.52 Function Mcl_TrgMuxEnableLock

This function enable lock of a TRGMUX register.

Details:

This function is used for enabling lock of a TRGMUX register

Return: void.

Prototype: `FUNC(void, MCL_CODE) Mcl_TrgMuxEnableLock(VAR(Mcl_TrgMuxRegisterIndexType, AUTOMATIC) registerIndex)`

Table 3-112. Mcl_TrgMuxEnableLock Arguments

Type	Name	Direction	Description
`Mcl_TrgMuxRegisterIndexType`	registerIndex	**input**	TRGMUX register index.

3.8.3.53 Function Mcl_Flexio_Enable

This function enable the Flexio feature.

Details:

Return: void.

Prototype: `void Mcl_Flexio_Enable(void);`

Table 3-113. Mcl_Flexio_Enable Arguments

Type	Name	Direction	Description
void		**input**	This function has no argument

3.8.3.54 Function Mcl_Flexio_Disable

This function disable the Flexio feature.

Details:

Return: void.

Prototype: `void Mcl_Flexio_Disable(void);`

Table 3-114. Mcl_Flexio_Disable Arguments

Type	Name	Direction	Description
void		**input**	This function has no argument

3.8.3.55 Function Mcl_Flexio_ClearShiftStat

This function clears bits from the Shifter Status register of FlexIO

Details: This function is reentrant and clears bits from the Shifter Status register of FlexIO * each bit from lsb to msb represents a channel from 0 to 7, and a "1" clears the bit * for the respective channel

Return: void

Prototype: `void Mcl_Flexio_ClearShiftStat(uint8 u8ShifterMask);`

Table 3-115. Mcl_Flexio_ClearShiftStat Arguments

Type	Name	Direction	Description
uint8	u8ShifterMask	**input**	8bit Mask of the shifter status flags to be cleared

3.8.3.56 Function Mcl_Flexio_ReadShiftStat

This function reads bits from the Shifter Status register of FlexIO

Details: This function is reentrant and reads bits from the Shifter Status register of FlexIO * each bit from lsb to msb represents a channel from 0 to 7, and a "1" clears the bit * for the respective channel

Return: 8 bit value of the Shift Status register & u8ShifterMask

Prototype: `uint8 Mcl_Flexio_ReadShiftStat(uint8 u8ShifterMask);`

Table 3-116. Mcl_Flexio_ReadShiftStat Arguments

Type	Name	Direction	Description
uint8	u8ShifterMask	**input**	8bit Mask of the shifter status flags to be read

3.8.3.57 Function Mcl_Flexio_ClearShiftErr

This function clears bits from the Shifter Error register of FlexIO

Details:

Return: void.

Prototype: `void Mcl_Flexio_ClearShiftErr(uint8 u8ShifterMask);`

Table 3-117. Mcl_Flexio_ClearShiftErr Arguments

Type	Name	Direction	Description
uint8	u8ShifterMask	**input**	8bit Mask of the shifter error flags to be cleared

3.8.3.58 Function Mcl_Flexio_ReadShiftErr

This function reads bits from the Shifter Error register of FlexIO.

Details:

Return: 8 bit value of the Shift Error register & u8ShifterMask.

Prototype: `uint8 Mcl_Flexio_ReadShiftErr(uint8 u8ShifterMask);`

Table 3-118. Mcl_Flexio_ReadShiftErr Arguments

Type	Name	Direction	Description
uint8	u8ShifterMask	**input**	8bit Mask of the shifter rrror flags to be read

3.8.3.59 Function Mcl_Flexio_ClearTimStat

This function clears bits from the Timer Status register of FlexIO.

Details:

Return: void

Prototype: `void Mcl_Flexio_ClearTimStat(uint8 u8TimerMask);`

Table 3-119. Mcl_Flexio_ClearTimStat Arguments

Type	Name	Direction	Description
uint8	u8TimerMask	**input**	8bit Mask of the timer status flags to be cleared

3.8.3.60 Function Mcl_Flexio_ReadTimStat

This function reads bits from the Timer Status register of FlexIO.

Details:

Return: 8 bit value of the Timer Status register & u8TimerMask

Prototype: `uint8 Mcl_Flexio_ReadTimStat(uint8 u8TimerMask);`

Table 3-120. Mcl_Flexio_ReadTimStat Arguments

Type	Name	Direction	Description
uint8	u8TimerMask	**input**	8bit Mask of the timer status flags to be cleared

3.8.3.61 Function Mcl_Flexio_WriteShiftSien

This function is used for writing in shifter status interrupt.

Details:

Return: void.

Prototype: `void Mcl_Flexio_WriteShiftSien(uint8 u8ShifterMask, uint8 u8ShifterEnableMask);`

Table 3-121. Mcl_Flexio_WriteShiftSien Arguments

Type	Name	Direction	Description
uint8	u8ShifterMask	**input**	8bit Mask of the shifter status interrupt to be write
uint8	u8ShifterEnableMask	**input**	The shifter's number will be writing in shifter status interrupt

3.8.3.62 Function Mcl_Flexio_ReadShiftSien

This function is used for reading in shifter status interrupt.

Details:

Return: Shifter Status Flag interrupt is enable or not (1 or 0).

Prototype: `uint8 Mcl_Flexio_ReadShiftSien(uint8 u8ShifterMask);`

Table 3-122. Mcl_Flexio_ReadShiftSien Arguments

Type	Name	Direction	Description
uint8	u8ShifterMask	**input**	8bit Mask of the shifter status interrupt to be read

3.8.3.63 Function Mcl_Flexio_WriteShiftEien

This function is used for writing in shifter error interrupt.

Details:

Return: void.

Prototype: `void Mcl_Flexio_WriteShiftEien(uint8 u8ShifterMask, uint8 u8ShifterEnableMask);`

Table 3-123. Mcl_Flexio_WriteShiftEien Arguments

Type	Name	Direction	Description
uint8	u8ShifterMask	**input**	8bit Mask of the shifter error interrupt to be write
uint8	u8ShifterEnableMask	**input**	The shifter's number will be writing in shifter error interrupt

3.8.3.64 Function Mcl_Flexio_ReadShiftEien

This function is used for reading in shifter error interrupt.

Details:

Return: Shifter Error Flag interrupt is enable or not (1 or 0).

Prototype: `uint8 Mcl_Flexio_ReadShiftEien(uint8 u8ShifterMask);`

Table 3-124. Mcl_Flexio_ReadShiftEien Arguments

Type	Name	Direction	Description
uint8	u8ShifterMask	**input**	8bit Mask of the shifter error interrupt to be read

3.8.3.65 Function Mcl_Flexio_WriteTimIen

This function is used for writing in Timer interrupt enable.

Details:

Return: void.

Prototype: `void Mcl_Flexio_WriteTimIen(uint8 u8TimerMask, uint8 u8TimerEnableMask);`

Table 3-125. Mcl_Flexio_WriteTimIen Arguments

Type	Name	Direction	Description
uint8	u8TimerMask	**input**	8bit Mask of the Timer interrupt enable to be write
uint8	u8TimerEnableMask	**input**	The shifter's number will be writing in Timer interrupt enable

3.8.3.66 Function Mcl_Flexio_ReadTimIen

This function is used for reading in Timer interrupt enable.

Details:

Return: Timer interrupt enable flag is enable or not (1 or 0).

Prototype: `uint8 Mcl_Flexio_ReadTimIen(uint8 u8TimerMask);`

Table 3-126. Mcl_Flexio_ReadTimIen Arguments

Type	Name	Direction	Description
uint8	u8TimerMask	**input**	8bit Mask of the Timer interrupt enable to be read

3.8.3.67 Function Mcl_Flexio_WriteShiftSden

This function is used for writing in shifter status DMA.

Details:

Return: void.

Prototype: `void Mcl_Flexio_WriteShiftSden(uint8 u8ShifterMask, uint8 u8ShifterEnableMask);`

Table 3-127. Mcl_Flexio_WriteShiftSden Arguments

Type	Name	Direction	Description
uint8	u8ShifterMask	**input**	8bit Mask of the shifter status DMA to be write
uint8	u8ShifterEnableMask	**input**	The shifter's number will be writing in shifter status DMA.

3.8.3.68 Function Mcl_Flexio_ReadShiftSden

This function is used for reading in shifter status DMA.

Details:

Return: shifter status DMA flag is enable or not (1 or 0).

Prototype: `uint8 Mcl_Flexio_ReadShiftSden(uint8 u8ShifterMask);`

Table 3-128. Mcl_Flexio_ReadShiftSden Arguments

Type	Name	Direction	Description
uint8	u8ShifterMask	**input**	8bit Mask of the shifter status DMA to be read

3.8.3.69 Function Mcl_Flexio_EnableInterrupts

This function is used for enable interrupt generation .

Details:

Return: void.

Prototype: `void Mcl_Flexio_EnableInterrupts(uint8 u8ShifterMask, uint8 u8ErrMask, uint8 u8TimerMask);`

Table 3-129. Mcl_Flexio_EnableInterrupts Arguments

Type	Name	Direction	Description
Mcl_ShifterType	u8ShifterMask	**input**	8bit Mask of the shifter interrupt to be enabled.
Mcl_ShifterType	u8ErrMask	**input**	8bit Mask of the shifter Error interrupt to be enabled.
Mcl_ShifterType	u8TimerMask	**input**	8bit Mask of the Timer interrupt to be enabled.

3.8.3.70 Function Mcl_Flexio_DisableInterrupts

This function is used for disable interrupt generation .

Details:

Return: void.

Prototype: `void Mcl_Flexio_DisableInterrupts(uint8 u8ShifterMask, uint8 u8ErrMask, uint8 u8TimerMask);`

Table 3-130. Mcl_Flexio_DisableInterrupts Arguments

Type	Name	Direction	Description
Mcl_ShifterType	u8ShifterMask	**input**	8bit Mask of the shifter interrupt to be disable.
Mcl_ShifterType	u8ErrMask	**input**	8bit Mask of the shifter Error interrupt to be disable.
Mcl_ShifterType	u8TimerMask	**input**	8bit Mask of the Timer interrupt to be disable.

3.8.3.71 Function Mcl_Flexio_SoftwareReset

This function is used for reset all configuration of Shifter, timer, pin.

Details:

Return: void.

Prototype: `void Mcl_Flexio_SoftwareReset(void);`

Table 3-131. Mcl_Flexio_SoftwareReset Arguments

Type	Name	Direction	Description
void		**input**	This function has no argument.

3.8.3.72 Function Mcl_SelectCommonTimebase

Implementation specific function to updates the Global Timebase bits of configured modules.

Details: This function is used to set the global timebase bits for modules that support the global timebase feature. The function selects the module that gives the common timebase and the modules that are use this timebase (as bits in u16ElementSyncList). Then it synchronizes the modules.

Return: void.

Prototype: `void Mcl_SelectCommonTimebase(uint8 ModuleId, uint16 u16ElementSyncList);`

Table 3-132. Mcl_SelectCommonTimebase Arguments

Type	Name	Direction	Description
uint8	ModuleId	**input**	Ftm module id
uint16	u16ElementSyncList	**input**	Ftm module mask value

3.8.4 Structs Reference

This chapter describes the structs supported by the MCL driver.

3.8.4.1 Structure Mcl_ChannelConfigType

Mcl Dma channel high level configuration structure.

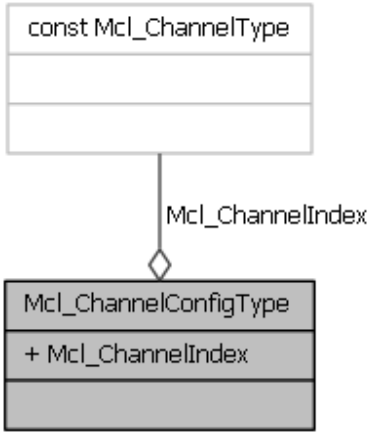

Figure 3-3. Struct Mcl_ChannelConfigType

Implements: Mcl_Dma_ChannelConfigType_struct

Declaration:

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

Table 3-133. Structure Mcl_ChannelConfigType member description

Member	Description
Mcl_ChannelIndex	Dma Channel configuration.

3.8.4.2 Structure Mcl_DmaInitConfigType

Mcl Dma high level configuration structure.

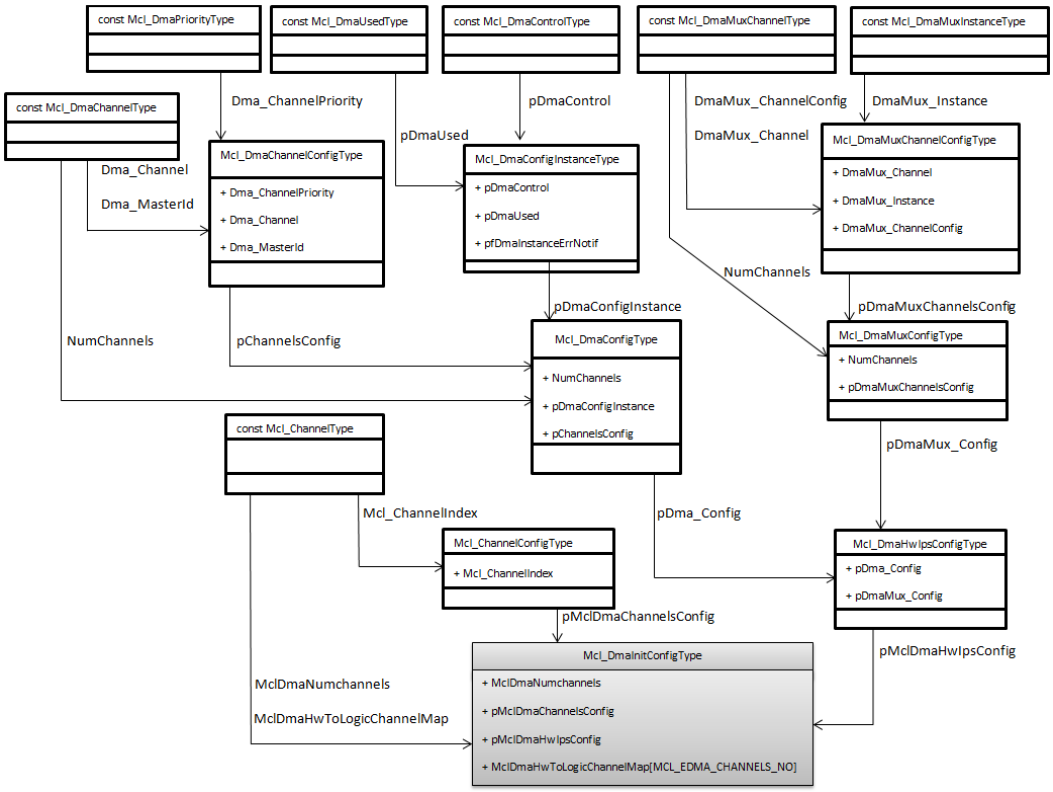

Figure 3-4. Struct Mcl_DmaInitConfigType

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 3-134. 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.

3.8.4.3 Structure Mcl_ConfigType

Mcl high level configuration structure.

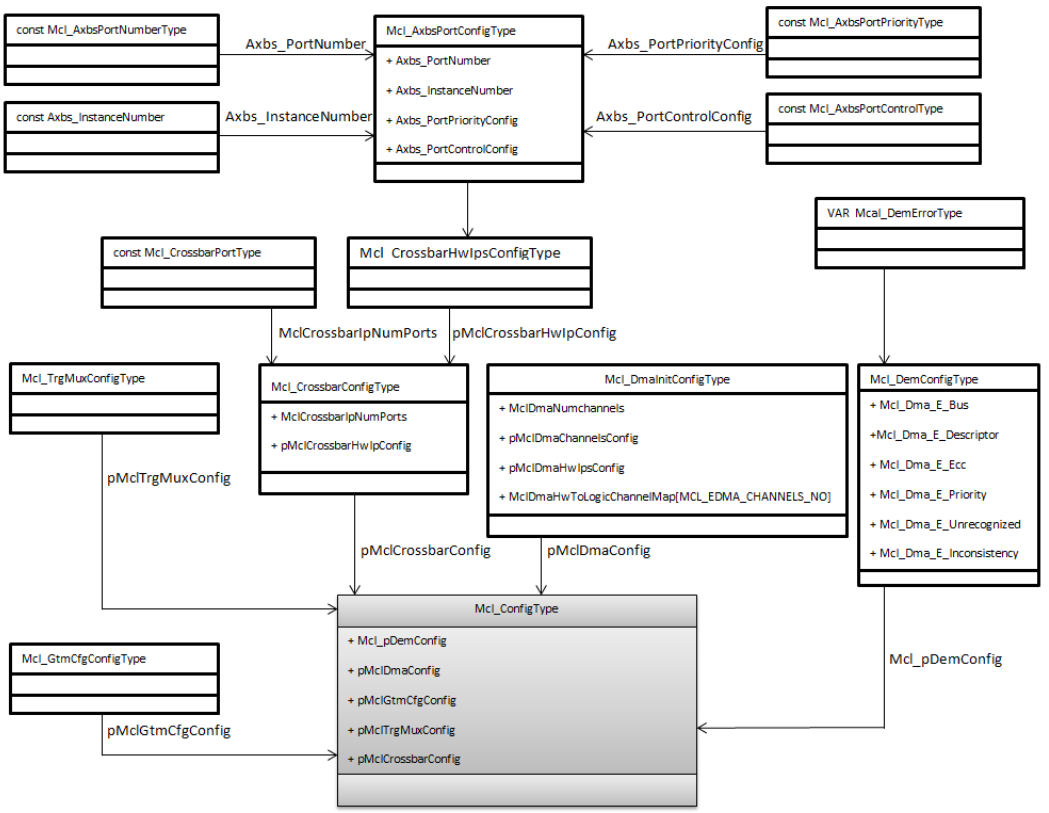

Figure 3-5. Struct Mcl_ConfigType

<u>Implements</u>: 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_TrgMuxConfigType * pMclTrgMuxConfig,
                                 #endif
                                 #if (MCL_ENABLE_CROSSBAR == STD_ON)
                                     const Mcl_CrossbarConfigType * pMclCrossbarConfig
                                 #endif
                                } Mcl_ConfigType;
```

Table 3-135. 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.

3.8.4.4 Structure Mcl_DmaChannelConfigType

Dma channel configuration structure.

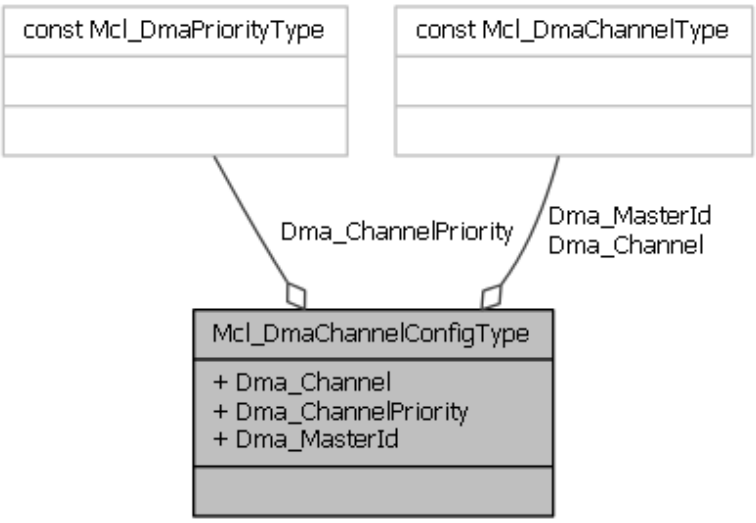

Figure 3-6. Struct Mcl_DmaChannelConfigType

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 3-136. Structure Mcl_DmaChannelConfigType member description

Member	Description
Dma_Channel	eDma channel used

Table continues on the next page...

Table 3-136. Structure Mcl_DmaChannelConfigType member description (continued)

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

3.8.4.5 Structure Mcl_DmaConfigType

Dma configuration structure.

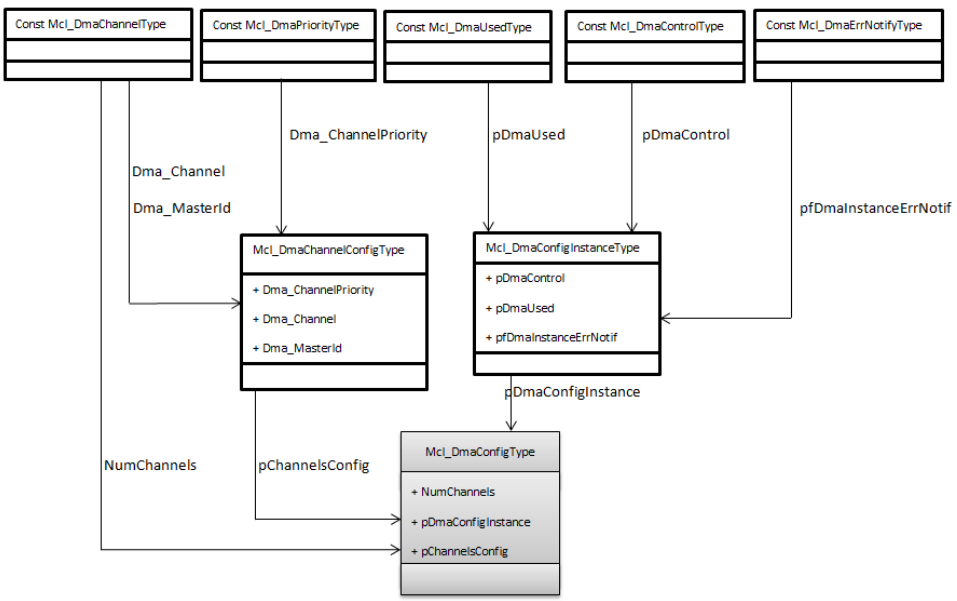

Figure 3-7. Struct Mcl_DmaConfigType

Implements: Mcl_DmaConfigType_struct

Declaration:

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

Table 3-137. 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.

3.8.4.6 Structure Mcl_DmaMuxChannelConfigType

DmaMux channel configuration structure.

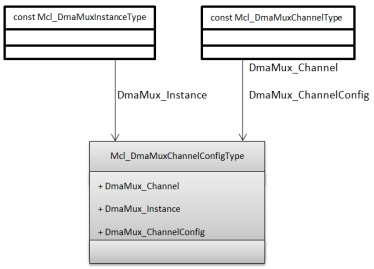

Figure 3-8. Struct Mcl_DmaMuxChannelConfigType

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 3-138. Structure Mcl_DmaMuxChannelConfigType member description

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

3.8.4.7 Structure Mcl_DmaMuxConfigType

DmaMux configuration structure.

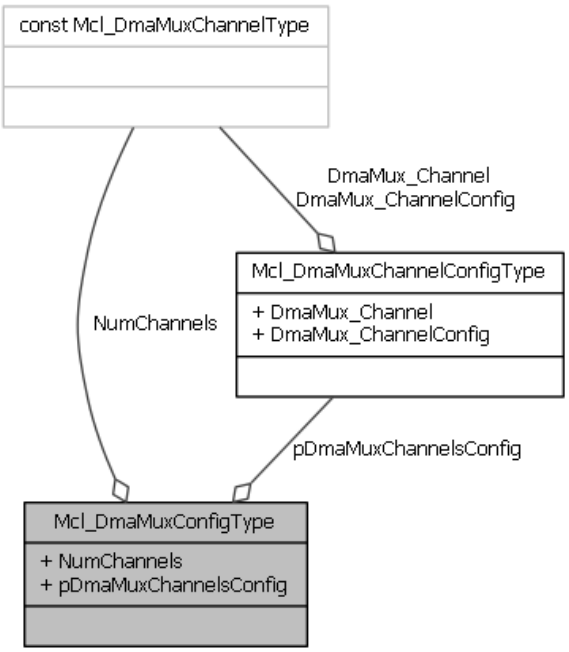

Figure 3-9. Struct Mcl_DmaMuxConfigType

Implements: DmaMux_ConfigType_struct

Declaration:

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

Table 3-139. Structure Mcl_DmaMuxConfigType member description

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

3.8.4.8 Structure Mcl_DmaTcdAttributesType

structure used for a basic configuration of a TCD

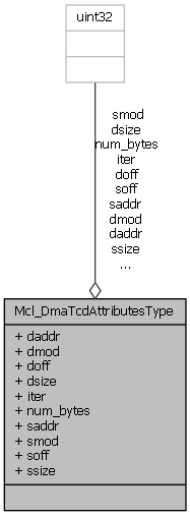

Figure 3-10. Struct Mcl_DmaTcdAttributesType

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 3-140. 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 3-140. Structure Mcl_DmaTcdAttributesType member description (continued)

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

3.8.4.9 Structure Mcl_DmaHwIpsConfigType

Mcl driver configuration structure.

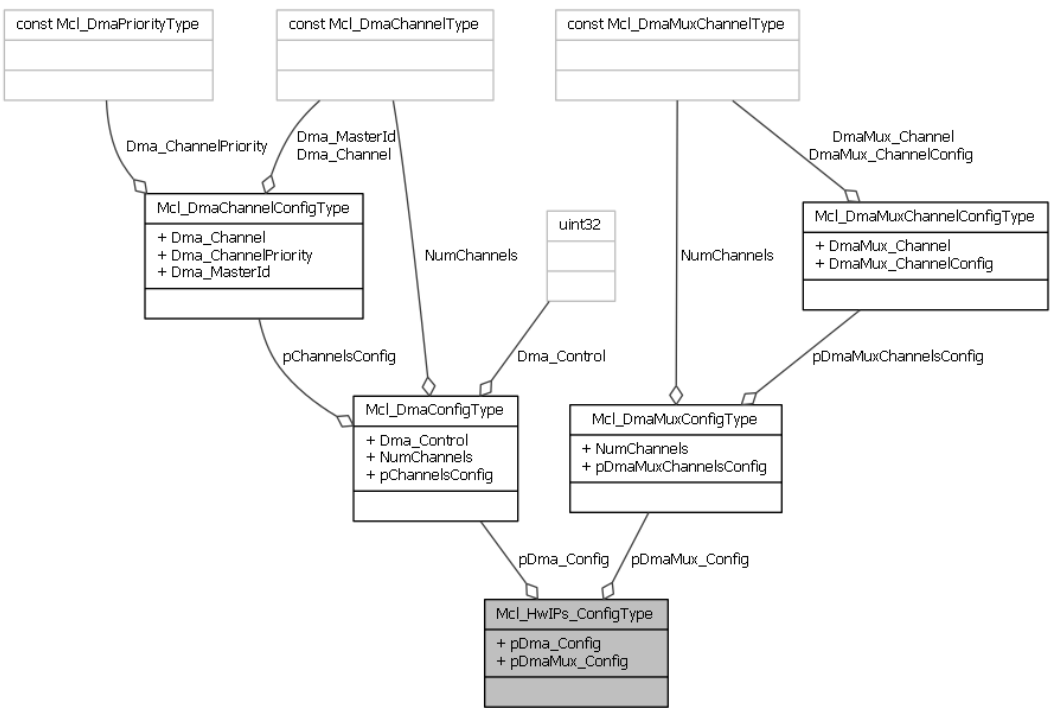

Figure 3-11. Struct Mcl_DmaHwIpsConfigType

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 3-141. 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.

3.8.4.10 Structure Mcl_DmaGlobalErrorStatusType

Dma channel configuration structure.

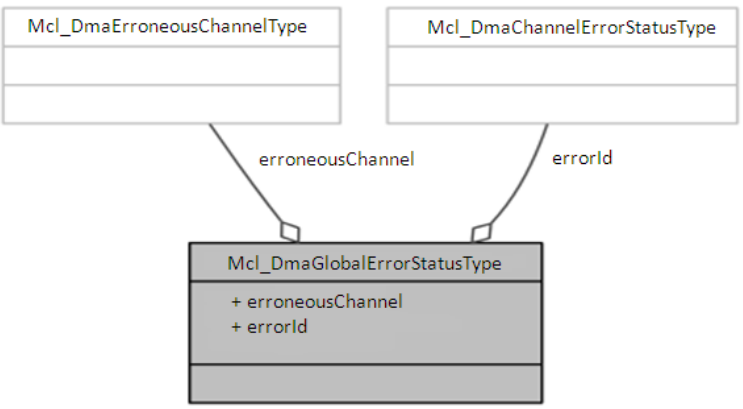

Figure 3-12. Struct Mcl_DmaGlobalErrorStatusType

Implements: Mcl_DmaGlobalErrorStatusType_struct

Declaration:

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

Table 3-142. Structure Mcl_DmaGlobalErrorStatusType member description

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

3.8.4.11 Structure Mcl_FlexioConfigType

FlexIO configuration structure.

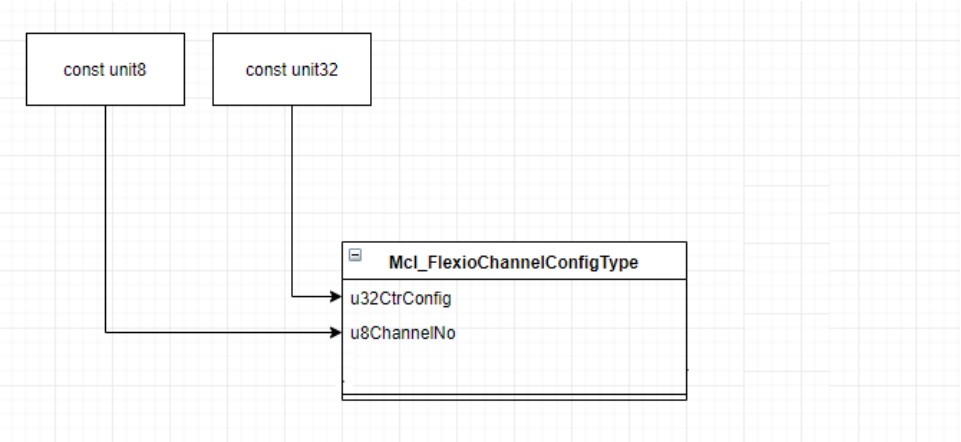

Figure 3-13. Struct Mcl_FlexioConfigType

Declaration:

```
typedef struct
                    {
                      const uint32 u32CtrConfig,
                      const uint8 u8ChannelNo
                    } Mcl_FlexioConfigType;
```

Table 3-143. Structure Mcl_FlexioConfigType member description

Member	Description
u32CtrConfig	Id of FlexIO application logic channel.
u8ChannelNo	Combination of physic shifters which are used for application logic channel.

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 specifing 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 S32K14X, 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)
```

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 S32K14X 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

3.9 Symbolic Names DISCLAIMER

All containers having the symbolic name tag set as true in the Autosar schema will generate defines like #define <Container_Short_Name> <Container_ID>

For this reason it is forbidden to duplicate the name of such containers across the MCAL configuration, or to use names that may trigger other compile issues (e.g. match existing #ifdefs arguments).

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
- MclIsrAvailable
- MclConfigSet
- CommonPublishedInformation

4.2 Form IMPLEMENTATION_CONFIG_VARIANT

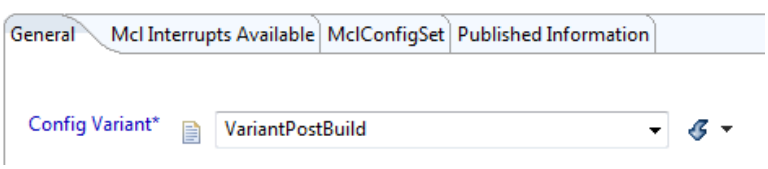

Figure 4-1. Tresos Plugin snapshot for IMPLEMENTATION_CONFIG_VARIANT form.

Table 4-1. Attribute IMPLEMENTATION_CONFIG_VARIANT detailed description

Property	Value
Label	Config Variant
Default	VariantPostBuild
Range	VariantPostBuild VariantPreCompile

4.3 Form McIGeneral

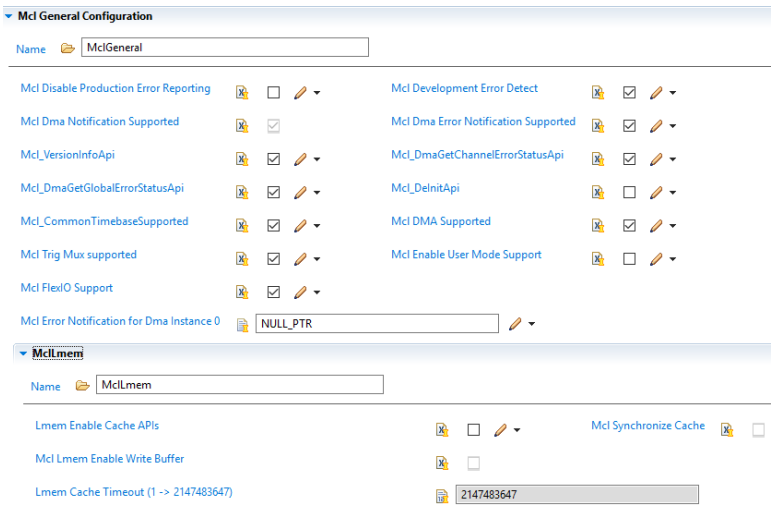

Figure 4-2. Tresos Plugin snapshot for McIGeneral form.

4.3.1 McIDisableDemReportErrorStatus (McIGeneral)

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

- true: Disabled.
- false: Enabled.

Table 4-2. Attribute McIDisableDemReportErrorStatus (McIGeneral) detailed description

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

4.3.2 McIDevErrorDetect (McIGeneral)

Switches the Development Error Detection and Notification on or off.

- true: Enabled.
- false: Disabled.

Table 4-3. Attribute McIDevErrorDetect (McIGeneral) detailed description

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

4.3.3 McIDmaNotificationSupported (McIGeneral)

Switches the Development Notification on or off.

- true: Enabled.
- false: Disabled.

Table 4-4. Attribute McIDmaNotificationSupported (McIGeneral) detailed description

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

4.3.4 McIErrorChecking (McIGeneral)

Switch to indicate if the error user notification is supported.

Table 4-5. Attribute McIErrorChecking (McIGeneral) detailed description

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

4.3.5 Mcl_VersionInfoApi (McIGeneral)

Table 4-6. Attribute Mcl_VersionInfoApi (McIGeneral) detailed description

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

4.3.6 Mcl_DmaGetChannelErrorStatusApi(McIGeneral)

Enables/Disables the get channel error status API Mcl_DmaGetChannelErrorStatus.

Table 4-7. Attribute Mcl_DmaGetChannelErrorStatusApi (McIGeneral) detailed description

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

4.3.7 Mcl_DmaGetGlobalErrorStatusApi(McIGeneral)

Enables/Disables the get global error status API Mcl_DmaGetGlobalErrorStatus.

Table 4-8. Attribute Mcl_DmaGetGlobalErrorStatusApi (McIGeneral) detailed description

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

4.3.8 Mcl_DeInitApi (MclGeneral)

Enables/Disables the Deinit API.

Table 4-9. Attribute Mcl_DeInitApi (MclGeneral) detailed description

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

4.3.9 EnableDMA (MclGeneral)

Enables/Disables the get global error status API Mcl_DmaGetGlobalErrorStatus.

Table 4-10. Attribute EnableDMA (MclGeneral) detailed description

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

4.3.10 Mcl_CommonTimebaseSupported (MclGeneral)

Enables/Disables the Mcl_CommonTimebaseSupported API.

Table 4-11. Attribute Mcl_CommonTimebaseSupported (MclGeneral) detailed description

Property	Value
Label	Mcl Common Time Base Support supported
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.3.11 McIEnableTrgMux (McIGeneral)

Activates/Deactivates Trigger mux configuration.

Table 4-12. Attribute McIEnableTrgMux (McIGeneral) detailed description

Property	Value
Label	Mcl Trig Mux supported
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.3.12 EnableFlexioSupport (McIGeneral)

Activates/Deactivates FlexIO configuration.

Table 4-13. Attribute EnableFlexioSupport (McIGeneral) detailed description

Property	Value
Label	Mcl FlexIO supported
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.3.13 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 4-14. Attribute McIErrorNotificationDma0 (McIGeneral) detailed description

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

4.3.14 MclLmemEnableCacheApi (MclGeneral)

Enables/Disables Cache APIs

Table 4-15. Attribute MclLmemEnableCacheApi (MclGeneral\MclLmem) detailed description

Property	Value
Label	Lmem Enable Cache APIs
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.3.15 MclSynchronizeCache (MclGeneral)

Enables/Disables Synchronize Cache

Table 4-16. Attribute MclSynchronizeCache (MclGeneral\MclLmem) detailed description

Property	Value
Label	Mcl Synchronize Cache
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.3.16 MclLmemEnableWriteBuffer (MclGeneral)

Enables/Disables Lmem Enable Write Buffer

Table 4-17. Attribute MclLmemEnableWriteBuffer (MclGeneral\MclLmem) detailed description

Property	Value
Label	Mcl Lmem Enable Write Buffer
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.3.17 MclLmemCacheTimeout (MclGeneral)

User to set timeout for executing Cache command

Table 4-18. Attribute MclLmemCacheTimeout (MclGeneral\MclLmem) detailed description

Property	Value
Label	Lmem Cache Timeout
Type	Integer
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	2147483647

4.4 Form MclDemEventParameterRefs

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.

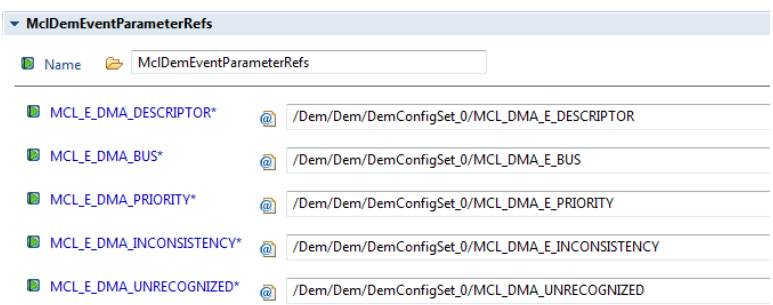

Figure 4-3. Tresos Plugin snapshot for MclDemEventParameterRefs form.

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 4-19. Attribute MCL_DMA_E_DESCRIPTOR (McIDemEventParameterRefs) detailed description

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

4.4.2 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 4-20. Attribute MCL_DMA_E_BUS (McIDemEventParameterRefs) detailed description

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

4.4.3 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 4-21. Attribute MCL_DMA_E_PRIORITY (McIDemEventParameterRefs) detailed description

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

4.4.4 MCL_DMA_E_INCONSISTENCY (McIDemEventParameterRefs)

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 4-22. Attribute MCL_DMA_E_INCONSISTENCY (MclDemEventParameterRefs) detailed description

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

4.4.5 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 4-23. Attribute MCL_DMA_E_UNRECOGNIZED (MclDemEventParameterRefs) detailed description

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

4.5 Form MclConfigSet

This container is the base for a multiple configuration set

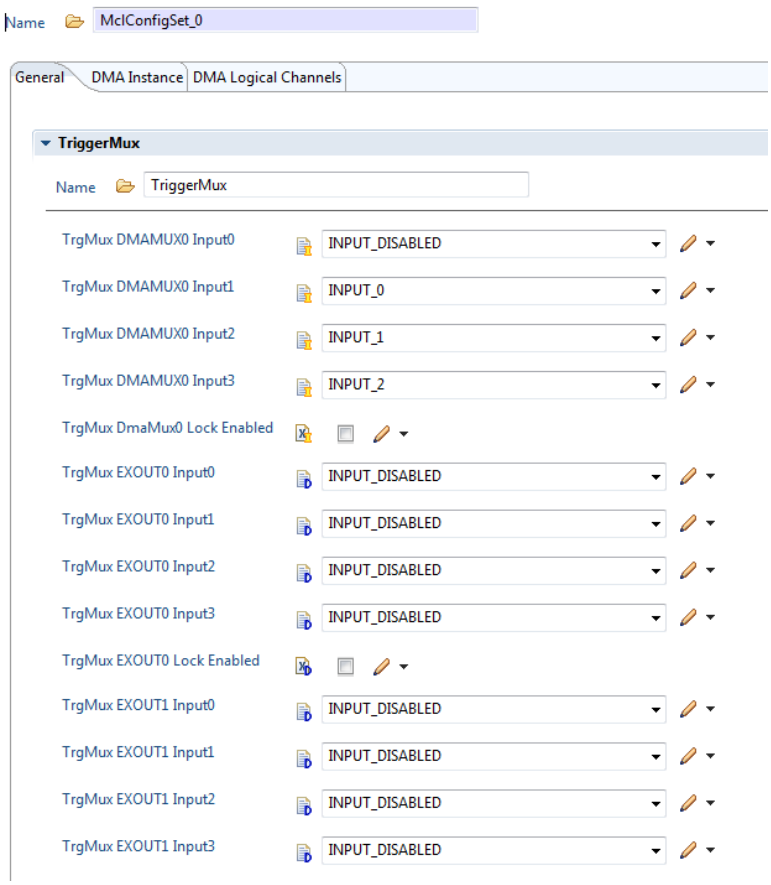

Figure 4-4. Tresos Plugin snapshot for McIConfigSet form.

4.5.1 Form DMATriggerMux

All data needed to configure trigger mux.

Is included by form : Form McIConfigSet

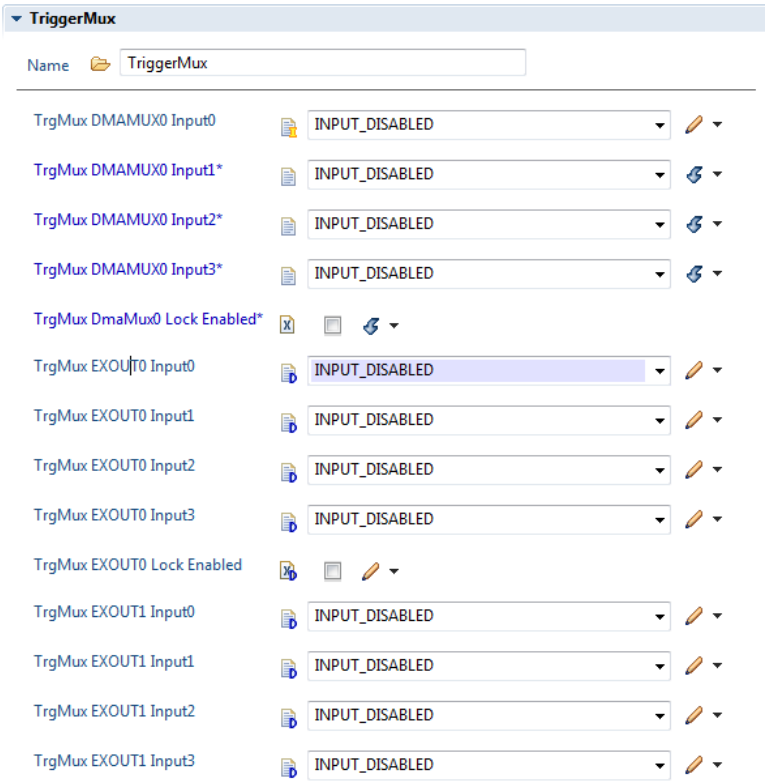

Figure 4-5. Tresos Plugin snapshot for DMATriggerMux form.

4.5.1.1 TrgMuxDmaMux0Input0

Used to configure the MUX select for peripheral trigger input 0

Table 4-24. Attribute TrgMuxDmaMux0Input0 detailed description

Property	Value
Label	TrgMux DMAMUX0 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.2 TrgMuxDmaMux0Input1

Used to configure the MUX select for peripheral trigger input 0

Table 4-25. Attribute TrgMuxDmaMux0Input1 detailed description

Property	Value
Label	TrgMux DMAMUX0 Input2
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.3 TrgMuxDmaMux0Input2

Used to configure the MUX select for peripheral trigger input 0

Table 4-26. Attribute TrgMuxDmaMux0Input2 detailed description

Property	Value
Label	TrgMux DMAMUX0 Input2
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.4 TrgMuxDmaMux0Input3

Used to configure the MUX select for peripheral trigger input 0

Table 4-27. Attribute TrgMuxDmaMux0Input3 detailed description

Property	Value
Label	TrgMux DMAMUX0 Input3
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.5 TrgMuxDmaMux0LockEn

Configures if register TrgMuxDmaMux0 must be locked(read-only).

Table 4-28. Attribute TrgMuxDmaMux0LockEn detailed description

Property	Value
Label	TrgMux DmaMux0 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.6 TrgMuxXOut0Input0

Used to configure the XbOut03 input 0

Table 4-29. Attribute TrgMuxXOut0Input0 detailed description

Property	Value
Label	TrgMux EXOUT0 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.7 TrgMuxXOut0Input1

Used to configure the XbOut03 input 1

Table 4-30. Attribute TrgMuxXOut0Input1 detailed description

Property	Value
Label	TrgMux EXOUT0 Input1
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.8 TrgMuxXOut0Input2

Used to configure the XbOut03 input 2

Table 4-31. Attribute TrgMuxXOut0Input2 detailed description

Property	Value
Label	TrgMux EXOUT0 Input2
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.9 TrgMuxXOut0Input0

Used to configure the XbOut03 input 3

Table 4-32. Attribute TrgMuxXOut0Input3 detailed description

Property	Value
Label	TrgMux EXOUT0 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.10 TrgMuxXOut0LockEn

Configures if register XbOut03 must be locked(read-only).

Table 4-33. Attribute TrgMuxXOut0LockEn detailed description

Property	Value
Label	TrgMux EXOUT0 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.11 TrgMuxXOut1Input0

Used to configure the XbOut47 input 0

Table 4-34. Attribute TrgMuxXOut1Input0 detailed description

Property	Value
Label	TrgMux EXOUT0 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.12 TrgMuxXOut1Input1

Used to configure the XbOut47 input 1

Table 4-35. Attribute TrgMuxXOut1Input1 detailed description

Property	Value
Label	TrgMux EXOUT0 Input1
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.13 TrgMuxXOut1Input2

Used to configure the XbOut47 input 2

Table 4-36. Attribute TrgMuxXOut1Input2 detailed description

Property	Value
Label	TrgMux EXOUT0 Input2
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.14 TrgMuxXOut1Input0

Used to configure the XbOut47 input 3

Table 4-37. Attribute TrgMuxXOut1Input3 detailed description

Property	Value
Label	TrgMux EXOUT0 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.15 TrgMuxXOut1LockEn

Configures if register XbOut47 must be locked(read-only).

Table 4-38. Attribute TrgMuxXOut1LockEn detailed description

Property	Value
Label	TrgMux EXOUT0 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.16 TrgMuxAdc0Input0

Used to configure the Adc0 input 0

Table 4-39. Attribute TrgMuxAdc0Input0 detailed description

Property	Value
Label	TrgMux ADC_0 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.17 TrgMuxAdc0Input1

Used to configure the Adc0 input 1

Table 4-40. Attribute TrgMuxAdc0Input1 detailed description

Property	Value
Label	TrgMux ADC_0 Input1
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.18 TrgMuxAdc0Input2

Used to configure the Adc0 input 2

Table 4-41. Attribute TrgMuxAdc0Input2 detailed description

Property	Value
Label	TrgMux ADC_0 Input2
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.19 TrgMuxAdc0Input3

Used to configure the Adc0 input 3

Table 4-42. Attribute TrgMuxAdc0Input3 detailed description

Property	Value
Label	TrgMux ADC_0 Input3
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.20 TrgMuxAdc0LockEn

Configures if register Adc0 must be locked(read-only).

Table 4-43. Attribute TrgMuxAdc0LockEn detailed description

Property	Value
Label	TrgMux ADC_0 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.21 TrgMuxAdc1Input0

Used to configure the Adc1 input 0

Table 4-44. Attribute TrgMuxAdc1Input0 detailed description

Property	Value
Label	TrgMux ADC_1 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.22 TrgMuxAdc1Input1

Used to configure the Adc1 input 1

Table 4-45. Attribute TrgMuxAdc1Input1 detailed description

Property	Value
Label	TrgMux ADC_1 Input1
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.23 TrgMuxAdc1Input2

Used to configure the Adc1 input 2

Table 4-46. Attribute TrgMuxAdc1Input2 detailed description

Property	Value
Label	TrgMux ADC_1 Input2
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.24 TrgMuxAdc1Input3

Used to configure the Adc1 input 3

Table 4-47. Attribute TrgMuxAdc1Input3 detailed description

Property	Value
Label	TrgMux ADC_1 Input3
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.25 TrgMuxAdc1LockEn

Configures if register Adc1 must be locked(read-only).

Table 4-48. Attribute TrgMuxAdc1LockEn detailed description

Property	Value
Label	TrgMux ADC_1 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.26 TrgMuxCmp0Input0

Used to configure the Cmp0 input 0.

Table 4-49. Attribute TrgMuxCmp0Input0 detailed description

Property	Value
Label	TrgMux CMP0 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.27 TrgMuxCmp0LockEn

Configures if register Cmp0 must be locked(read-only).

Table 4-50. Attribute TrgMuxCmp0LockEn detailed description

Property	Value
Label	TrgMux CMP0 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.28 TrgMuxFtm0Input0

Used to configure Ftm0 input 0

Table 4-51. Attribute TrgMuxFtm0Input0 detailed description

Property	Value
Label	TrgMux FTM0 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.29 TrgMuxFtm0Input1

Used to configure Ftm0 input 1

Table 4-52. Attribute TrgMuxFtm0Input1 detailed description

Property	Value
Label	TrgMux FTM0 Input1
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.30 TrgMuxFtm0Input2

Used to configure Ftm0 input 2

Table 4-53. Attribute TrgMuxFtm0Input2 detailed description

Property	Value
Label	TrgMux FTM0 Input2
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.31 TrgMuxFtm0Input3

Used to configure Ftm0 input 3

Table 4-54. Attribute TrgMuxFtm0Input3 detailed description

Property	Value
Label	TrgMux FTM0 Input3
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.32 TrgMuxFtm0LockEn

Configures if register FTM0 must be locked(read-only).

Table 4-55. Attribute TrgMuxFtm0LockEn detailed description

Property	Value
Label	TrgMux FTM0 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.33 TrgMuxFtm1Input0

Used to configure Ftm1 input 0

Table 4-56. Attribute TrgMuxFtm1Input0 detailed description

Property	Value
Label	TrgMux FTM1 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.34 TrgMuxFtm1Input1

Used to configure Ftm1 input 1

Table 4-57. Attribute TrgMuxFtm1Input1 detailed description

Property	Value
Label	TrgMux FTM1 Input1
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.35 TrgMuxFtm1Input2

Used to configure Ftm1 input 2

Table 4-58. Attribute TrgMuxFtm1Input2 detailed description

Property	Value
Label	TrgMux FTM1 Input2
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.36 TrgMuxFtm1Input3

Used to configure Ftm1 input 3

Table 4-59. Attribute TrgMuxFtm1Input3 detailed description

Property	Value
Label	TrgMux FTM1 Input3
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.37 TrgMuxFtm1LockEn

Configures if register FTM1 must be locked(read-only).

Table 4-60. Attribute TrgMuxFtm1LockEn detailed description

Property	Value
Label	TrgMux FTM1 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.38 TrgMuxFtm3Input0

Used to configure Ftm3 input 0

Table 4-61. Attribute TrgMuxFtm3Input0 detailed description

Property	Value
Label	TrgMux Ftm3 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.39 TrgMuxFtm3Input1

Used to configure Ftm3 input 1

Table 4-62. Attribute TrgMuxFtm3Input1 detailed description

Property	Value
Label	TrgMux Ftm3 Input1
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.40 TrgMuxFtm3Input2

Used to configure Ftm3 input 2

Table 4-63. Attribute TrgMuxFtm3Input2 detailed description

Property	Value
Label	TrgMux Ftm3 Input2
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.41 TrgMuxFtm3Input3

Used to configure Ftm3 input 3

Table 4-64. Attribute TrgMuxFtm3Input3 detailed description

Property	Value
Label	TrgMux Ftm3 Input3
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.42 TrgMuxFtm3LockEn

Configures if register Ftm3 must be locked(read-only).

Table 4-65. Attribute TrgMuxFtm3LockEn detailed description

Property	Value
Label	TrgMux Ftm3 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.43 TrgMuxPdb0Input0

Used to configure the Pdb0input 0.

Table 4-66. Attribute TrgMuxPdb0Input0 detailed description

Property	Value
Label	TrgMux PDB0 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.44 TrgMuxPdb0LockEn

Configures if register Pdb0 must be locked(read-only).

Table 4-67. Attribute TrgMuxPdb0LockEn detailed description

Property	Value
Label	TrgMux PDB0 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.45 TrgMuxPdb1Input0

Used to configure the Pdb1input 0.

Table 4-68. Attribute TrgMuxPdb1Input0 detailed description

Property	Value
Label	TrgMux PDB1 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.46 TrgMuxPdb1LockEn

Configures if register Pdb1 must be locked(read-only).

Table 4-69. Attribute TrgMuxPdb1LockEn detailed description

Property	Value
Label	TrgMux PDB1 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.47 TrgMuxFlexIoInput0

Used to configure FlexIo input 0

Table 4-70. Attribute TrgMuxFlexIoInput0 detailed description

Property	Value
Label	TrgMux FLEXIO Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.48 TrgMuxFlexIoInput1

Used to configure FlexIo input 1

Table 4-71. Attribute TrgMuxFlexIoInput1 detailed description

Property	Value
Label	TrgMux FLEXIO Input1
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.49 TrgMuxFlexIoInput2

Used to configure FlexIo input 2

Table 4-72. Attribute TrgMuxFlexIoInput2 detailed description

Property	Value
Label	TrgMux FLEXIO Input2
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.50 TrgMuxFlexIoInput3

Used to configure FlexIo input 3

Table 4-73. Attribute TrgMuxFlexIoInput3 detailed description

Property	Value
Label	TrgMux FLEXIO Input3
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.51 TrgMuxFlexIoLockEn

Configures if register FlexIo must be locked(read-only).

Table 4-74. Attribute TrgMuxFlexIoLockEn detailed description

Property	Value
Label	TrgMux FLEXIO Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.52 TrgMuxLpitInput0

Used to configure Lpit input 0

Table 4-75. Attribute TrgMuxLpitInput0 detailed description

Property	Value
Label	TrgMux LPIT Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.53 TrgMuxLpitInput1

Used to configure Lpit input 1

Table 4-76. Attribute TrgMuxLpitInput1 detailed description

Property	Value
Label	TrgMux LPIT Input1
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.54 TrgMuxLpitInput2

Used to configure Lpit input 2

Table 4-77. Attribute TrgMuxLpitInput2 detailed description

Property	Value
Label	TrgMux LPIT Input2
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.55 TrgMuxLpitInput3

Used to configure Lpit input 3

Table 4-78. Attribute TrgMuxLpitInput3 detailed description

Property	Value
Label	TrgMux LPIT Input3
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.56 TrgMuxLpitLockEn

Configures if register Lpit must be locked(read-only).

Table 4-79. Attribute TrgMuxLpitLockEn detailed description

Property	Value
Label	TrgMux LPIT Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.57 TrgMuxLpuart0Input0

Used to configure the Lpuart0 input 0

Table 4-80. Attribute TrgMuxLpuart0Input0 detailed description

Property	Value
Label	TrgMux LPUART0 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.58 TrgMuxLpuart0LockEn

Configures if register Lpuart0 must be locked(read-only).

Table 4-81. Attribute TrgMuxLpuart0LockEn detailed description

Property	Value
Label	TrgMux LPUART0 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.59 TrgMuxLpuart1Input0

Used to configure the Lpuart1 input 0

Table 4-82. Attribute TrgMuxLpuart1Input0 detailed description

Property	Value
Label	TrgMux LPUART1 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.60 TrgMuxLpuart1LockEn

Configures if register Lpuart1 must be locked(read-only).

Table 4-83. Attribute TrgMuxLpuart1LockEn detailed description

Property	Value
Label	TrgMux LPUART1 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.61 TrgMuxLpi2c0Input0

Used to configure the Lpi2c0 input 0

Table 4-84. Attribute TrgMuxLpi2c0Input0 detailed description

Property	Value
Label	TrgMux LPI2C0 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.62 TrgMuxLpi2c0LockEn

Configures if register Lpi2c0 must be locked(read-only).

Table 4-85. Attribute TrgMuxLpi2c0LockEn detailed description

Property	Value
Label	TrgMux LPI2C0 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.63 TrgMuxLpspi0Input0

Used to configure the Lpspi0 input 0

Table 4-86. Attribute TrgMuxLpspi0Input0 detailed description

Property	Value
Label	TrgMux LPSPI0 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.64 TrgMuxLpspi0LockEn

Configures if register Lpspi0 must be locked(read-only).

Table 4-87. Attribute TrgMuxLpspi0LockEn detailed description

Property	Value
Label	TrgMux LPSPI0 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.65 TrgMuxLpspi1Input0

Used to configure the Lpspi1 input 0

Table 4-88. Attribute TrgMuxLpspi1Input0 detailed description

Property	Value
Label	TrgMux LPSPI1 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.66 TrgMuxLpspi1LockEn

Configures if register Lpspi1 must be locked(read-only).

Table 4-89. Attribute TrgMuxLpspi1LockEn detailed description

Property	Value
Label	TrgMux LPSPI1 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.67 TrgMuxLptmr0Input0

Used to configure the Lptmr0 input 0

Table 4-90. Attribute TrgMuxLptmr0Input0 detailed description

Property	Value
Label	TrgMux LPTMR0 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.68 TrgMuxLptmr0LockEn

Configures if register Lptmr0 must be locked(read-only).

Table 4-91. Attribute TrgMuxLptmr0LockEn detailed description

Property	Value
Label	TrgMux LPTMR0 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.69 TrgMuxLpi2c1Input0

Used to configure the Lpi2c1 input 0

Table 4-92. Attribute TrgMuxLpi2c1Input0 detailed description

Property	Value
Label	TrgMux LPI2C1 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.70 TrgMuxLpi2c1LockEn

Configures if register Lpi2c1 must be locked(read-only).

Table 4-93. Attribute TrgMuxLpi2c1LockEn detailed description

Property	Value
Label	TrgMux LPI2C1 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.71 TrgMuxFtm4Input0

Used to configure the Ftm4 input 0

Table 4-94. Attribute TrgMuxFtm4Input0 detailed description

Property	Value
Label	TrgMux FTM4 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.72 TrgMuxFtm4LockEn

Configures if register Ftm4 must be locked(read-only).

Table 4-95. Attribute TrgMuxFtm4LockEn detailed description

Property	Value
Label	TrgMux FTM4 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.73 TrgMuxFtm5Input0

Used to configure the Ftm5 input 0

Table 4-96. Attribute TrgMuxFtm5Input0 detailed description

Property	Value
Label	TrgMux FTM5 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.74 TrgMuxFtm5LockEn

Configures if register Ftm5 must be locked(read-only).

Table 4-97. Attribute TrgMuxFtm5LockEn detailed description

Property	Value
Label	TrgMux FTM5 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.75 TrgMuxFtm6Input0

Used to configure the Ftm6 input 0

Table 4-98. Attribute TrgMuxFtm6Input0 detailed description

Property	Value
Label	TrgMux FTM6 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.76 TrgMuxFtm6LockEn

Configures if register Ftm6 must be locked(read-only).

Table 4-99. Attribute TrgMuxFtm6LockEn detailed description

Property	Value
Label	TrgMux FTM6 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.1.77 TrgMuxFtm7Input0

Used to configure the Ftm7 input 0

Table 4-100. Attribute TrgMuxFtm7Input0 detailed description

Property	Value
Label	TrgMux FTM7 Input0
Type	String(Range)
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	INPUT_DISABLED

4.5.1.78 TrgMuxFtm7LockEn

Configures if register Ftm7 must be locked(read-only).

Table 4-101. Attribute TrgMuxFtm7LockEn detailed description

Property	Value
Label	TrgMux FTM7 Lock Enabled
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

4.5.2 Form FlexioConfig

Is included by form : Form MclConfigSet

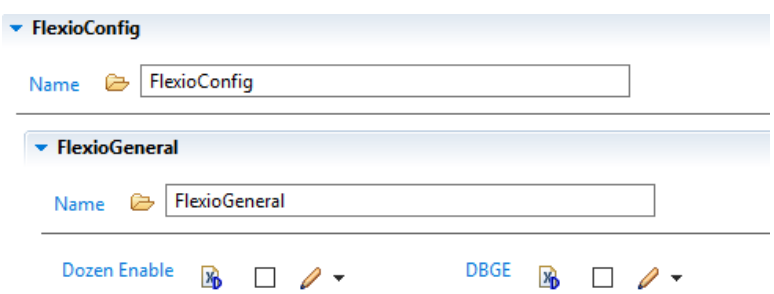

Figure 4-6. Tresos Plugin snapshot for FlexioConfig form.

4.5.2.1 DozenEnable

FLEXIO_CTRL[DOZEN].Doze Enable.0 - FlexIO enabled in Doze modes.1 - FlexIO disabled in Doze modes.

Table 4-102. Attribute DozenEnable detailed description

Property	Value
Label	Dozen Enable
Type	Boolean
Origin	NXP
Symbolic Name	false
Default	false

4.5.2.2 DBGE

FLEXIO_CTRL[DBGE].Debug Enable.0 - FlexIO is disabled in debug modes.1 - FlexIO disabled in debug modes.

Table 4-103. Attribute DBGE detailed description

Property	Value
Label	DBGE
Type	Boolean
Origin	NXP
Symbolic Name	false
Default	false

4.5.3 Form DMAInstance

All data needed to configure one DMA Instance.

Is included by form : Form MclConfigSet

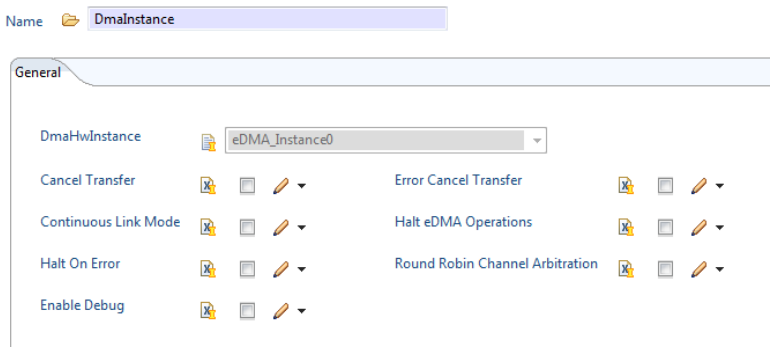

Figure 4-7. Tresos Plugin snapshot for DMAInstance form.

4.5.3.1 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 4-104. Attribute McIEDMA_CX (McIConfigSet) detailed description

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

4.5.3.2 McIEDMA_ECX (McIConfigSet)

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 4-105. Attribute McIEDMA_ECX (McIConfigSet) detailed description

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

4.5.3.3 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 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 4-106. Attribute McIEDMA_CLM (McIConfigSet) detailed description

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

4.5.3.4 McIEDMA_HALT (McIConfigSet)

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 4-107. Attribute McIEDMA_HALT (McIConfigSet) detailed description

Property	Value
Label	Halt eDMA Operations
Type	BOOLEAN
Origin	Custom

Table continues on the next page...

Table 4-107. Attribute McIEDMA_HALT (McIConfigSet) detailed description (continued)

Property	Value
Symbolic Name	false
Default	false

4.5.3.5 McIEDMA_HOE (McIConfigSet)

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 4-108. Attribute McIEDMA_HOE (McIConfigSet) detailed description

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

4.5.3.6 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 4-109. Attribute McIEDMA_ERCA (McIConfigSet) detailed description

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

4.5.3.7 MclEDMA_EDBG (MclConfigSet)

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 4-110. Attribute MclEDMA_EDBG (MclConfigSet) detailed description

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

4.5.4 Form DMAChannel

All data needed to configure one DMA channel.

Is included by form : Form MclConfigSet

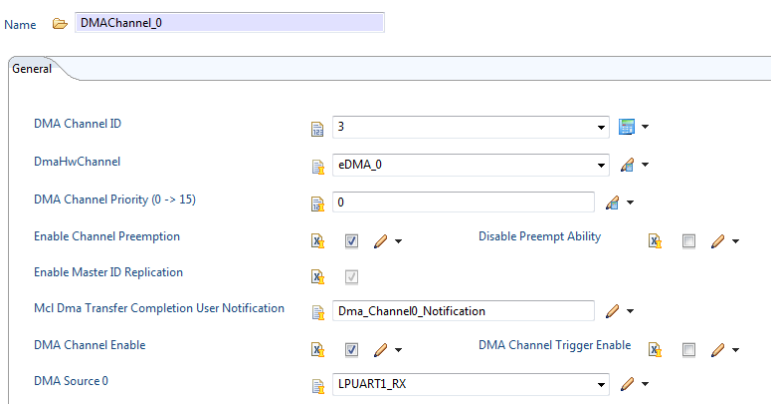

Figure 4-8. Tresos Plugin snapshot for DMAChannel form.

4.5.4.1 MclDMAChannelId (DMAChannel)

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

Table 4-111. Attribute McIDMAChannelId (DMAChannel) detailed description

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

4.5.4.2 DmaHwChannel (DMAChannel)

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

Table 4-112. Attribute DmaHwChannel (DMAChannel) detailed description

Property	Value
Type	ENUMERATION
Origin	Custom
Symbolic Name	false

4.5.4.3 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 4-113. Attribute DMAChannelPriority (DMAChannel) detailed description

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

4.5.4.4 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 4-114. Attribute ECP (DMAChannel) detailed description

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

4.5.4.5 DPA (DMAChannel)

Disable preemptive ability.

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

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

Table 4-115. Attribute DPA (DMAChannel) detailed description

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

4.5.4.6 EMI (DMAChannel)

Enable Master ID replication.

0 (unchecked) - Master ID replication is disabled

1 (checked) - Master ID replication is enabled

Table 4-116. Attribute EMI (DMAChannel) detailed description

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

4.5.4.7 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 4-117. 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

4.5.4.8 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 4-118. Attribute McIDMAChannelEnable (DMAChannel) detailed description

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

4.5.4.9 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 4-119. Attribute McIDMAChannelTriggerEnable (DMAChannel) detailed description

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

4.5.4.10 DmaSource0 (DMAChannel)

Configuration for DMA source slot in DmaMux0 (Physical DMA channels from 0 to 15) NOTE: This is an Implementation Specific Parameter.

Table 4-120. Attribute DmaSource0 (DMAChannel) detailed description

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

4.6 Form McIIsrAvailable

Mcl Interrupts Available

Index	Name	Mcl Interr...	Mcl Interr...
0	McIIsrAvail...	eDMA_0	
1	McIIsrAvail...	eDMA_1	
2	McIIsrAvail...	eDMA_10	
3	McIIsrAvail...	eDMA_11	
4	McIIsrAvail...	eDMA_12	
5	McIIsrAvail...	eDMA_13	
6	McIIsrAvail...	eDMA_14	
7	McIIsrAvail...	eDMA_15	
8	McIIsrAvail...	eDMA_2	
9	McIIsrAvail...	eDMA_3	
10	McIIsrAvail...	eDMA_4	
11	McIIsrAvail...	eDMA_5	
12	McIIsrAvail...	eDMA_6	
13	McIIsrAvail...	eDMA_7	
14	McIIsrAvail...	eDMA_8	
15	McIIsrAvail...	eDMA_9	

Figure 4-9. Tresos Plugin snapshot for McIIsrAvailable form.

4.6.1 McIIsrName (McIIsrAvailable)

Mcl Interrupt Name.

Table 4-121. Attribute McIIsrName (McIIsrAvailable) detailed description

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

4.6.2 McIIsrEnabled (McIIsrAvailable)

Switch to indicate if the interrupt is enabled.

- true: Enabled.
- false: Disabled.

Table 4-122. Attribute McIIsrEnabled (McIIsrAvailable) detailed description

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

4.7 Form CommonPublishedInformation

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

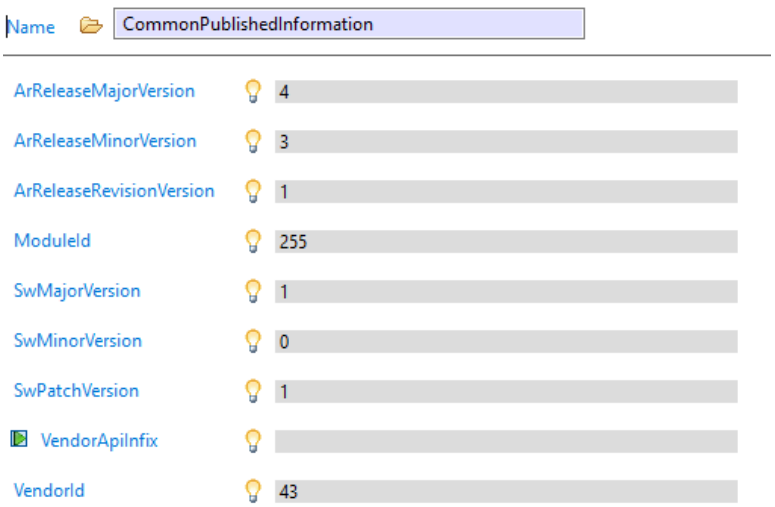

Figure 4-10. Tresos Plugin snapshot for CommonPublishedInformation form.

4.7.1 ArReleaseMajorVersion (CommonPublishedInformation)

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

Table 4-123. Attribute ArReleaseMajorVersion (CommonPublishedInformation) detailed description

Property	Value
Label	AUTOSAR Major Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	4
Invalid	Range >=4 <=4

4.7.2 ArReleaseMinorVersion (CommonPublishedInformation)

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

Table 4-124. Attribute ArReleaseMinorVersion (CommonPublishedInformation) detailed description

Property	Value
Label	AUTOSAR Minor Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	3
Invalid	Range >=3 <=3

4.7.3 ArReleaseRevisionVersion (CommonPublishedInformation)

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

Table 4-125. Attribute ArReleaseRevisionVersion (CommonPublishedInformation) detailed description

Property	Value
Label	AUTOSAR Release Revision Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	1
Invalid	Range >=1 <=1

4.7.4 ModuleId (CommonPublishedInformation)

Module ID of this module from Module List.

Table 4-126. Attribute ModuleId (CommonPublishedInformation) detailed description

Property	Value
Label	Module Id
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	255
Invalid	Range >=255 <=255

4.7.5 SwMajorVersion (CommonPublishedInformation)

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

Table 4-127. 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

4.7.6 SwMinorVersion (CommonPublishedInformation)

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

Table 4-128. Attribute SwMinorVersion (CommonPublishedInformation) detailed description

Property	Value
Label	Software Minor Version
Type	INTEGER_LABEL

Table continues on the next page...

Table 4-128. Attribute SwMinorVersion (CommonPublishedInformation) detailed description (continued)

Property	Value
Origin	Custom
Symbolic Name	false
Default	0
Invalid	Range >=0 <=0

4.7.7 SwPatchVersion (CommonPublishedInformation)

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

Table 4-129. Attribute SwPatchVersion (CommonPublishedInformation) detailed description

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

4.7.8 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 4-130. Attribute VendorApiInfix (CommonPublishedInformation) detailed description

Property	Value
Label	Vendor Api Infix
Type	STRING_LABEL
Origin	Custom
Symbolic Name	false
Default	
Enable	false

4.7.9 VendorId (CommonPublishedInformation)

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

Table 4-131. 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

Web Support:
nxp.com/support

Document Number UM2MCLASR4.3 Rev0001R1.0.1
Revision 1.0