MCAL Configuration Verification Manual for Dma

32-bit TriCore™ AURIX™ TC3xx microcontroller family

About this document

Scope and purpose

This Configuration Data Reference document is applicable to all TC3xx devices in the TriCore™ AURIX™ family of 32-bit microcontrollers.

The purpose of this document is to facilitate the integrator to verify the generated code based on the input configuration parameters. This document describes details of structures, defines, macros and variables generated from the configuration parameters.

Intended audience

This document is intended for integrators who need to understand the logic of the generated configuration code of AURIX™ AUTOSAR MCAL.

Reference documents

This document should be read in conjunction with the following documents:

- AURIX™ TC3xx MCAL User Manual Dma

Table of contents

About this document **1**
Table of contents **2**
1 Dma driver **4**
1.1 File: Dma_Cfg.h 4
1.1.1 Macro: DMA_AR_RELEASE_MAJOR_VERSION 4
1.1.2 Macro: DMA_AR_RELEASE_MINOR_VERSION 4
1.1.3 Macro: DMA_AR_RELEASE_REVISION_VERSION 4
1.1.4 Macro: DMA_SW_MAJOR_VERSION 5
1.1.5 Macro: DMA_SW_MINOR_VERSION 5
1.1.6 Macro: DMA_SW_PATCH_VERSION 5
1.1.7 Macro: DMA_SAFETY_ENABLE 6
1.1.8 Macro: DMA_INITCHECK_API 6
1.1.9 Macro: DMA_RUNTIME_API_MODE 6
1.1.10 Macro: DMA_INIT_API_MODE 7
1.1.11 Macro: DMA_DEV_ERROR_DETECT 7
1.1.12 Macro: DMA_MULTICORE_ERROR_DETECT 7
1.1.13 Macro: DMA_MAX_NUM_OF_CHANNELS 8
1.1.14 Macro: DMA_CHDEINIT_API 8
1.1.15 Macro: DMA_SUSPEND_API 8
1.1.16 Macro: DMA_TRIGGER_API 9
1.1.17 Macro: DMA_DEINIT_API 9
1.1.18 Macro: DMA_SETPATTERN_API 9
1.1.19 Macro: DMA_DATA_PENDING_API 10
1.1.20 Macro: DMA_BUFFER_SWITCH_API 11
1.1.21 Macro: DMA_GETVERSIONINFO_API 11
1.1.22 Macro: DMA_MAX_TRANSACTION_SET_PER_CHANNEL 11
1.1.23 Macro: DMA_LINKED_LIST_ENABLE 12
1.1.24 Macro: DMA_DOUBLE_BUFFER_ENABLE 12
1.1.25 Macro: DMA_NUM_OF_CHANNELS 13
1.1.26 Macro: DMA_ALLOCATED_CHANNELS_CORE<x> 13
1.2 File: Dma[_<variant>]_PBcfg.c 15
1.2.1 Structure: Dma_Config [_<variant>] 15
1.2.1.1 Member: DmaCoreConfigPtr[MCAL_NO_OF_CORES] 19
1.2.1.2 Member: DmaMovEngErr [2] 20
1.2.1.3 Member: DmaHwResourcePartition 21
1.2.1.4 Member: Dma_ChannelMaps[DMA_MAX_NUM_OF_CHANNELS] 22
1.2.1.5 Member: DmaChTrlEnabled [DMA_MAX_NUM_OF_CHANNELS] 23
1.2.1.6 Member: DmaTotalChConfigured 24
1.2.2 Structure: Dma_ChConfigRootCore<x> 24
1.2.2.1 Member: DmaChConfigPtr 25
1.2.2.2 Member: DmaNumberofChConfiguredPerCore 25
1.2.3 Structure: Dma_kChConfigRoot_Core<x> 26
1.2.3.1 Member: DmaChTCSPtr 27
1.2.3.2 Member: DmaChNotifPtr 28
1.2.3.3 Member: DmaTsrConfig 28
1.2.3.4 Member: DmaChNumber 29
1.2.3.5 Member: DmaChHwPartitionConfig 29

Table of contents

1.2.3.6 Member: DmaMEErrorNotifPtr 29
1.2.4 Structure: Dma_kChannel_<x>_TcsConfigRoot 30
1.2.4.1 Member: DmaReadDataCrc 31
1.2.4.2 Member: DmaSourceDestAddressCrc 32
1.2.4.3 Member: DmaSourceAddress 32
1.2.4.4 Member: DmaDestinationAddress 34
1.2.4.5 Member: DmaAddressInterruptControl 36
1.2.4.6 Member: DmaChannelConfig 39
1.2.4.7 Member: DmaShadowAddress 40
1.2.4.8 Member: DmaChControlStatus 43
1.2.5 Function declaration: Dma_ChannelNotificationPtrType 44
1.2.6 Function declaration: Dma_MoveEngineErrorNotificationPtrType 44
1.3 File: Dma[_<variant>]_PBcfg.h 45
1.3.1 Extern: Dma_Config [_<variant>] 45
Revision history 46

1 Dma driver

This chapter describes the details of the configuration data generated from the DMA driver.

1.1 File: Dma_Cfg.h

The generated header file contains all pre-compile configuration parameters. Pre-compile time configuration allows decoupling of the static configuration from implementation. The file is generated in 'inc' folder.

1.1.1 Macro: DMA_AR_RELEASE_MAJOR_VERSION

Table 1 DMA_AR_RELEASE_MAJOR_VERSION

Name	DMA_AR_RELEASE_MAJOR_VERSION	
Description	Major version number of AUTOSAR release on which the Dma implementation is based on.	
Verification method	The macro is generated with the value present in 'CommonPublishedInformation/ArMajorVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Dma_Cfg.h file with ArMajorVersion 4	`#define DMA_AR_RELEASE_MAJOR_VERSION (4U)`

1.1.2 Macro: DMA_AR_RELEASE_MINOR_VERSION

Table 2 DMA_AR_RELEASE_MINOR_VERSION

Name	DMA_AR_RELEASE_MINOR_VERSION	
Description	Minor version number of AUTOSAR release on which the Dma implementation is based on.	
Verification method	The macro is generated with the value present in 'CommonPublishedInformation/ArMinorVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Dma_Cfg.h file with ArMinorVersion 2	`#define DMA_AR_RELEASE_MINOR_VERSION (2U)`

1.1.3 Macro: DMA_AR_RELEASE_REVISION_VERSION

Table 3 DMA_AR_RELEASE_REVISION_VERSION

Name	DMA_AR_RELEASE_REVISION_VERSION

Description	Revision version number of AUTOSAR release on which the Dma implementation is based on.	
Verification method	The macro is generated with the value present in 'CommonPublishedInformation/ArPatchVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Dma_Cfg.h file with ArPatchVersion 2	`#define DMA_AR_RELEASE_REVISION_VERSION (2U)`

1.1.4 Macro: DMA_SW_MAJOR_VERSION

Table 4 DMA_SW_MAJOR_VERSION

Name	DMA_SW_MAJOR_VERSION	
Description	Major version number of the Dma module.	
Verification method	The macro is generated with the value present in 'CommonPublishedInformation/SwMajorVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Dma_Cfg.h file with SwMajorVersion 10	`#define DMA_SW_MAJOR_VERSION (10U)`

1.1.5 Macro: DMA_SW_MINOR_VERSION

Table 5 DMA_SW_MINOR_VERSION

Name	DMA_SW_MINOR_VERSION	
Description	Minor version number of the Dma module.	
Verification method	The macro is generated with the value present in 'CommonPublishedInformation/SwMinorVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Dma_Cfg.h file with SwMinorVersion 10	`#define DMA_SW_MINOR_VERSION (10U)`

1.1.6 Macro: DMA_SW_PATCH_VERSION

Table 6 DMA_SW_PATCH_VERSION

Name	DMA_SW_PATCH_VERSION
Description	Patch level version number of the Dma module.
Verification method	The macro is generated with the value present in

	'CommonPublishedInformation/SwPatchVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Dma_Cfg.h file with SwPatchVersion 0	`#define DMA_SW_PATCH_VERSION (0U)`

1.1.7 Macro: DMA_SAFETY_ENABLE

Table 7 DMA_SAFETY_ENABLE

Name	DMA_SAFETY_ENABLE	
Description	Enables/disables safety features	
Verification method	The macro is generated as STD_ON if DmaSafetyEnable configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	DmaSafetyEnable = True	`#define DMA_SAFETY_ENABLE (STD_ON)`
	DmaSafetyEnable = False	`#define DMA_SAFETY_ENABLE (STD_OFF)`

1.1.8 Macro: DMA_INITCHECK_API

Table 8 DMA_INITCHECK_API

Name	DMA_INITCHECK_API	
Description	Enables/disables Dma_InitCheck API	
Verification method	The macro is generated as STD_ON if DmaInitCheckApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	DmaInitCheckApi = True	`#define DMA_INITCHECK_API (STD_ON)`
	DmaInitCheckApi = False	`#define DMA_INITCHECK_API (STD_OFF)`

1.1.9 Macro: DMA_RUNTIME_API_MODE

Table 9 DMA_RUNTIME_API_MODE

Name	DMA_RUNTIME_API_MODE	
Description	Decides the mode of execution of Run Time API's	
Verification method	The macro is generated as DMA_MCAL_USER1MODE if DmaRuntimeApiMode configuration parameter is set to 'DMA_MCAL_USER1MODE' else the macro is generated as DMA_MCAL_SUPERVISORMODE.	
Example(s)	**Action**	**Generated output**
	DmaRuntimeApiMode =	`#define DMA_RUNTIME_API_MODE`

	DMA_MCAL_USER1MODE	(DMA_MCAL_USER1MODE)
	DmaRuntimeApiMode = DMA_MCAL_SUPERVISORMODE	#define DMA_RUNTIME_API_MODE (DMA_MCAL_SUPERVISORMODE)

1.1.10 Macro: DMA_INIT_API_MODE

Table 10 DMA_INIT_API_MODE

Name	DMA_INIT_API_MODE	
Description	Decides the mode of execution of Init API's.	
Verification method	The macro is generated as DMA_MCAL_USER1MODE if DmaInitApiMode configuration parameter is set to 'DMA_MCAL_USER1MODE' else the macro is generated as DMA_MCAL_SUPERVISORMODE.	
Example(s)	**Action**	**Generated output**
	DmaInitApiMode = DMA_MCAL_USER1MODE	#define DMA_INIT_API_MODE (DMA_MCAL_USER1MODE)
	DmaInitApiMode = DMA_MCAL_SUPERVISORMODE	#define DMA_INIT_API_MODE (DMA_MCAL_SUPERVISORMODE)

1.1.11 Macro: DMA_DEV_ERROR_DETECT

Table 11 DMA_DEV_ERROR_DETECT

Name	DMA_DEV_ERROR_DETECT	
Description	Enables/disables the Development Error Detection.	
Verification method	The macro is generated as STD_ON if DmaDevErrorDetect configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	DmaDevErrorDetect = True	#define DMA_DEV_ERROR_DETECT (STD_ON)
	DmaDevErrorDetect = False	#define DMA_DEV_ERROR_DETECT (STD_OFF)

1.1.12 Macro: DMA_MULTICORE_ERROR_DETECT

Table 12 DMA_MULTICORE_ERROR_DETECT

Name	DMA_MULTICORE_ERROR_DETECT	
Description	Enables/disables Multicore DET Check	
Verification method	The macro is generated as STD_ON if DmaMultiCoreErrorDetect configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	DmaMultiCoreErrorDetect =	#define DMA_MULTICORE_ERROR_DETECT

	True	(STD_ON)
	DmaMultiCoreErrorDetect = False	#define DMA_MULTICORE_ERROR_DETECT (STD_OFF)

1.1.13 Macro: DMA_MAX_NUM_OF_CHANNELS

Table 13 DMA_MAX_NUM_OF_CHANNELS

Name	DMA_MAX_NUM_OF_CHANNELS	
Description	The number of DMA channels present in the microcontroller.	
Verification method	The macro is generated with the value based on the maximum DMA channels available in the hardware.	
Example(s)	**Action**	**Generated output**
	Maximum DMA channels = 127	#define DMA_MAX_NUM_OF_CHANNELS (127U)
	Maximum DMA channels = 63	#define DMA_MAX_NUM_OF_CHANNELS (63U)

1.1.14 Macro: DMA_CHDEINIT_API

Table 14 DMA_CHDEINIT_API

Name	DMA_CHDEINIT_API	
Description	Enables/disables Dma_ChDeInit API.	
Verification method	The macro is generated as STD_ON if DmaDeinitApiConfiguration parameter is set to ‘True’ else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	DmaDeinitApiConfiguration = True	#define DMA_CHDEINIT_API (STD_ON)
	DmaDeinitApiConfiguration = False	#define DMA_CHDEINIT_API (STD_OFF)

1.1.15 Macro: DMA_SUSPEND_API

Table 15 DMA_SUSPEND_API

Name	DMA_SUSPEND_API	
Description	Enable/disable the following APIs: Dma_ChTransferFreeze Dma_ChTransferResume.	
Verification method	The macro is generated as STD_ON if DmaSuspendApiConfiguration parameter is set to ‘True’ else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**

	DmaSuspendApiConfiguration = True	`#define DMA_SUSPEND_API (STD_ON)`
	DmaSuspendApiConfiguration = False	`#define DMA_SUSPEND_API (STD_OFF)`

1.1.16 Macro: DMA_TRIGGER_API

Table 16 DMA_TRIGGER_API

Name	DMA_TRIGGER_API	
Description	Enable/disable the following APIs: Dma_ChEnableHardwareTrigger Dma_ChDisableHardwareTrigger.	
Verification method	The macro is generated as STD_ON if DmaTriggerApiConfiguration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	DmaTriggerApiConfiguration = True	`#define DMA_TRIGGER_API (STD_ON)`
	DmaTriggerApiConfiguration = False	`#define DMA_TRIGGER_API (STD_OFF)`

1.1.17 Macro: DMA_DEINIT_API

Table 17 DMA_DEINIT_API

Name	DMA_DEINIT_API	
Description	Enable/disable the following API: Dma_DeInit.	
Verification method	The macro is generated as STD_ON if DmaDeInitApi parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	DmaDeInitApi = True	`#define DMA_DEINIT_API (STD_ON)`
	DmaDeInitApi = False	`#define DMA_DEINIT_API (STD_OFF)`

1.1.18 Macro: DMA_SETPATTERN_API

Table 18 DMA_SETPATTERN_API

Name	DMA_SETPATTERN_API	
Description	Enable/disable the pattern detection feature.	
Verification method	The macro is generated as STD_ON if DmaPatternMode parameter is enable or DmaTcsShadowRegisterConfiguration parameter is set to conditional linked-list else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	• Configur 'Pattern_Mode_1' in DmaPatternMode parameter (in the	`#define DMA_SETPATTERN_API (STD_ON)`

	'DmaChannelTransactionSet' container) • Configure 'DMA_CONDITIONAL_LINKED_LIST' in the 'DmaTcsShadowRegisterConfiguration' parameter (in the 'DmaChannelTransactionSet' container)	
	• Configur 'Pattern_Detection_Disabled' in DmaPatternMode parameter (in the 'DmaChannelTransactionSet' container) • Configure 'DMA_CONDITIONAL_LINKED_LIST' in the 'DmaTcsShadowRegisterConfiguration' parameter (in the 'DmaChannelTransactionSet' container)	`#define DMA_SETPATTERN_API (STD_ON)`
	• Configur 'Pattern_Detection_Disabled' in DmaPatternMode parameter (in the 'DmaChannelTransactionSet' container) • Configure 'DMA_SHADOWING_DISABLED' in the 'DmaTcsShadowRegisterConfiguration' parameter (in the 'DmaChannelTransactionSet' container)	`#define DMA_SETPATTERN_API (STD_OFF)`

1.1.19 Macro: DMA_DATA_PENDING_API

Table 19 DMA_DATA_PENDING_API

Name	DMA_DATA_PENDING_API	
Description	Enables/disables the Dma_ChGetRemainingData API.	
Verification method	The macro is generated as STD_ON if DmaDataPendingApiConfiguration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	DmaDataPendingApiConfiguration = True	`#define DMA_DATA_PENDING_API (STD_ON)`
	DmaDataPendingApiConfiguration = False	`#define DMA_DATA_PENDING_API`

		(STD_OFF)

1.1.20 Macro: DMA_BUFFER_SWITCH_API

Table 20 DMA_BUFFER_SWITCH_API

Name	DMA_BUFFER_SWITCH_API	
Description	Enables/disables Dma_ChSwitchBuffer API	
Verification method	The macro is generated as STD_ON if DmaBufferSwitchApiConfiguration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	DmaBufferSwitchApiConfiguration = True	`#define DMA_BUFFER_SWITCH_API (STD_ON)`
	DmaBufferSwitchApiConfiguration = False	`#define DMA_BUFFER_SWITCH_API (STD_OFF)`

1.1.21 Macro: DMA_GETVERSIONINFO_API

Table 21 DMA_GETVERSIONINFO_API

Name	DMA_GETVERSIONINFO_API	
Description	Enables/disables Dma_GetVersionInfo API	
Verification method	The macro is generated as STD_ON if DmaVersionInfoApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	DmaVersionInfoApi = True	`#define DMA_GETVERSIONINFO_API (STD_ON)`
	DmaVersionInfoApi = False	`#define DMA_GETVERSIONINFO_API (STD_OFF)`

1.1.22 Macro: DMA_MAX_TRANSACTION_SET_PER_CHANNEL

Table 22 DMA_MAX_TRANSACTION_SET_PER_CHANNEL

Name	DMA_MAX_TRANSACTION_SET_PER_CHANNEL	
Description	The maximum number of transaction sets allowed for a DMA channel, if the linked list feature is used.	
Verification method	The macro is generated with the value configured for the parameter '/DmaGeneral/DmaMaxTransactionSetPerChannel' in the configuration tool.	
Example(s)	**Action**	**Generated output**
	DmaMaxTransactionSetPerChannel = 150	`#define DMA_MAX_TRANSACTION_SET_PER_CHANNEL (150U)`

1.1.23 Macro: DMA_LINKED_LIST_ENABLE

Table 23 DMA_LINKED_LIST_ENABLE

Name	DMA_LINKED_LIST_ENABLE	
Description	Enables/disables the linked list feature.	
Verification method	The macro is generated as STD_ON if the parameter 'DmaChannelConfig/DmaChannelTransactionSet/DmaTcsShadowRegisterConfiguration' is configured to either: 'DMA_LINKED_LIST' or 'DMA_ACCUMULATED_LINKED_LIST' or 'DMA_SAFE_LINKED_LIST' or 'DMA_CONDITIONAL_LINKED_LIST', for any of the channels configured in the 'DmaChannelConfig' container.	
Example(s)	**Action**	**Generated output**
	• Configure one DMA channel (in the 'DmaChannelConfig' container) • Configure 'DMA_LINKED_LIST' in the 'DmaTcsShadowRegisterConfiguration' parameter.	`#define DMA_LINKED_LIST_ENABLE (STD_ON)`
	• Configure one DMA channel (in the 'DmaChannelConfig' container) • Configure 'DMA_SHADOWING_DISABLED' in the 'DmaTcsShadowRegisterConfiguration' parameter.	`#define DMA_LINKED_LIST_ENABLE (STD_OFF)`

1.1.24 Macro: DMA_DOUBLE_BUFFER_ENABLE

Table 24 DMA_DOUBLE_BUFFER_ENABLE

Name	DMA_DOUBLE_BUFFER_ENABLE	
Description	Enables/disables the double buffer feature.	
Verification method	The macro is generated as STD_ON if the parameter 'DmaChannelConfig/DmaChannelTransactionSet/DmaTcsShadowRegisterConfiguration' is configured to either: 'DMA_SOURCE_DOUBLE_BUFFERING_SW_SWITCH' or 'DMA_SOURCE_DOUBLE_BUFFERING_HW_SW_SWITCH' or 'DMA_DEST_DOUBLE_BUFFERING_SW_SWITCH' or 'DMA_DEST_DOUBLE_BUFFERING_HW_SW_SWITCH', for any of the channels configured in the ' DmaChannelConfig' container.	
Example(s)	**Action**	**Generated output**
	• Configure one DMA channel (in	`#define DMA_DOUBLE_BUFFER_ENABLE`

	the 'DmaChannelConfig' container) • Configure 'DMA_SOURCE_DOUBLE_BUFFERING_SW_SWITCH' in the 'DmaTcsShadowRegisterConfiguration' parameter.	(STD_ON)
	• Configure one DMA channel (in the 'DmaChannelConfig' container) • Configure 'DMA_SHADOWING_DISABLED' in the 'DmaTcsShadowRegisterConfiguration' parameter.	#define DMA_DOUBLE_BUFFER_ENABLE (STD_OFF)

1.1.25 Macro: DMA_NUM_OF_CHANNELS

Table 25 DMA_NUM_OF_CHANNELS

Name	DMA_NUM_OF_CHANNELS	
Description	The total number of channels configured in the configuration tool.	
Verification method	The macro is generated with the total number of channels which are configured in the 'DmaChannelConfig' container.	
Example(s)	**Action**	**Generated output**
	• Configure 5 DMA channels.	#define DMA_NUM_OF_CHANNELS (5U)

1.1.26 Macro: DMA_ALLOCATED_CHANNELS_CORE<x>

Table 26 DMA_ALLOCATED_CHANNELS_CORE<x>

Name	DMA_ALLOCATED_CHANNELS_CORE<x> (x ranges from 0 to 5)	
Description	Indicates the total number of channels configured for CORE<x>.	
Verification method	The macro is generated as total number of channels configured in the 'DmaChannelConfig' container that are allocated to CORE<x>. *Note: Channels not assigned to any core are assigned to master core (ResourceM/ResourceMMcalConfig/ResourceMMasterCore).*	
Example(s)	**Action**	**Generated output**
	• Configure 4 DMA channels (Channel0 to Channel3 in the 'DmaChannelConfig' container) • Set ResourceMMasterCore as	#define DMA_ALLOCATED_CHANNELS_CORE0 (0U) #define DMA_ALLOCATED_CHANNELS_CORE1 (4U)

	CORE1 (in the parameter 'ResourceM/ResourceMMcalConfig/ResourceMMasterCore') • Do not assign DMA channels in any ResourceMAllocation (in the container 'ResourceM/ResourceMMcalConfig/ResourceMMcalCore/ResourceMAllocation')	#define DMA_ALLOCATED_CHANNELS_CORE2 (0U) #define DMA_ALLOCATED_CHANNELS_CORE3 (0U) #define DMA_ALLOCATED_CHANNELS_CORE4 (0U) #define DMA_ALLOCATED_CHANNELS_CORE5 (0U)
	• Configure 9 DMA channels (Channel0 to Channel8 in the 'DmaChannelConfig' container). • Set ResourceMMasterCore as CORE2 (in the parameter 'ResourceM/ResourceMMcalConfig/ResourceMMasterCore') • Assign Channel0, Channel3 and Channel7 under ResourceMAllocation with ResourceMCoreID as CORE0. (in the container 'ResourceM/ResourceMMcalConfig/ResourceMMcalCore/ResourceMAllocation') • Assign Channel1, Channel2 and Channel8 under ResourceMAllocation with ResourceMCoreID as CORE2(in the container 'ResourceM/ResourceMMcalConfig/ResourceMMcalCore/ResourceMAllocation')	#define DMA_ALLOCATED_CHANNELS_CORE0 (3U) #define DMA_ALLOCATED_CHANNELS_CORE1 (0U) #define DMA_ALLOCATED_CHANNELS_CORE2 (6U) #define DMA_ALLOCATED_CHANNELS_CORE3 (0U) #define DMA_ALLOCATED_CHANNELS_CORE4 (0U) #define DMA_ALLOCATED_CHANNELS_CORE5 (0U)
	• Configure 4 DMA channels (Channel0 to Channel3 in the 'DmaChannelConfig' container). • Assign all the channels under ResourceMAllocation with ResourceMCoreID as CORE0 (in the container 'ResourceM/ResourceMMcalConfig/ResourceMMcalCore/ResourceMAllocation')	#define DMA_ALLOCATED_CHANNELS_CORE0 (4U) #define DMA_ALLOCATED_CHANNELS_CORE1 (0U) #define DMA_ALLOCATED_CHANNELS_CORE2 (0U) #define DMA_ALLOCATED_CHANNELS_CORE3 (0U) #define DMA_ALLOCATED_CHANNELS_CORE4 (0U) #define DMA_ALLOCATED_CHANNELS_CORE5 (0U)
	• Configure 9 DMA channels (Channel0 to Channel8 in the 'DmaChannelConfig' container).	#define DMA_ALLOCATED_CHANNELS_CORE0 (0U) #define

	• Set ResourceMMasterCore as CORE4 (in the parameter 'ResourceM/ResourceMMcalConfig/ResourceMMasterCore')
	• Assign Channel0, Channel3 and Channel7 under ResourceMAllocation with ResourceMCoreID as CORE3 (in the container 'ResourceM/ResourceMMcalConfig/ResourceMMcalCore/ResourceMAllocation')
		`DMA_ALLOCATED_CHANNELS_CORE1 (0U)`
		`#define DMA_ALLOCATED_CHANNELS_CORE2 (0U)`
		`#define DMA_ALLOCATED_CHANNELS_CORE3 (3U)`
		`#define DMA_ALLOCATED_CHANNELS_CORE4 (6U)`
		`#define DMA_ALLOCATED_CHANNELS_CORE5 (0U)`

1.2 File: Dma[_<variant>]_PBcfg.c

The generated file contains all post-build configuration parameters. Post-build time configuration mechanism allows configurable functionality of DMA driver that is deployed as object code. The file is generated in 'src' folder.

1.2.1 Structure: Dma_Config [_<variant>]

Table 27 Dma_Config [_<variant>]

Name	Dma_Config[_<variant>]	
Type	Dma_ConfigType	
Description	Root configuration structure of DMA driver which will be used during initialization.	
Verification method	The generated structure is present in Dma[_<variant>]_PBcfg.c file. The <variant> indicates the name of the post-build variant. For a variant-aware configuration the structure name is appended with the variant name. For variant-unaware configuration <variant> is ignored.	
Example(s)	**Action**	**Generated output**

	Configure the DMA channels (in 'DmaChannelConfig' container) indicated below (variant-unaware):
	• 3 Dma channels (Channel 0, 2, 4) to Core0
	• 2 Dma channels (Channel 6, 10) to Core 1
	
	Configure DMA channel and pattern for pattern matching feature (in DmaPatternConfig container)
	• Channel 2 in DmaChannelForPattern0 parameter
	• Channel 4 in DmaChannelForPattern0 parameter
	• Pattern 0x45 in DmaPattern0 parameter
		`const Dma_ConfigType Dma_Config=`
		`{`
		`  {`
		`  /* Channel Configuration root for Core 0*/`
		`  /* This is the number of resources:3 */`
		`  &Dma_ChConfigRootCore0,`
		`  /* Channel Configuration root for Core 1*/`
		`  /* This is the number of resources:2 */`
		`  &Dma_ChConfigRootCore1,`
		`  /* Channel Configuration root for Core 2*/`
		`  /* This is the number of resources:0 */`

• Pattern 0x54 in DmaPattern1 parameter	NULL_PTR, /* Channel Configuration root for Core 3*/ /* This is the number of resources:0 */ NULL_PTR, /* Channel Configuration root for Core 4*/ /* This is the number of resources:0 */ NULL_PTR, /* Channel Configuration root for Core 5*/ /* This is the number of resources:0 */ NULL_PTR, }, /* MoveEngine Error config for ME0 and ME1 */ {0X04030000U,0X04030000U}, /* pattern matching register value PRR0 and PRR1 */ { 0X45 , 0X54 } , /* Dma ChannelId for pattern0 and pattern1 */ {2 , 4}, /* Access partition configuration */ { /* Bus master configuration */ {0XFFFFFFFFU,0XFFFFFFFFU,0XFFFFFFFFU,0XFFFFFFFFU}, /* Resource partition configuration */ { 0X01U, 0X01U, 0X01U, 0X01U} }, /* { Channel Position Index, Channel Core Map } - The mapping data of channels */ {

	{0, 0 }, /* Channel 0 */
	{255, 255 }, /* Channel 1 */
	{1, 0 }, /* Channel 2 */
	{255, 255 }, /* Channel 3 */
	{2, 0 }, /* Channel 4 */
	{255, 255 }, /* Channel 5 */
	{0, 1 }, /* Channel 6 */
	{255, 255 }, /* Channel 7 */
	{255, 255 }, /* Channel 8 */
	{255, 255 }, /* Channel 9 */
	{1, 1 }, /* Channel 10 */
	{255, 255 }, /* Channel 11 */
	{255, 255 }, /* Channel 12 */
	.
	.
	.
	{255, 255 }, /* Channel 126 */
	{255, 255 }, /* Channel 127 */
	},
	{
	/* Bit map of the channels with TRL enabled */
	0x0U,
	0x0U,
	0x0U,
	0x0U,
	},
	/* Total number of DMA channels */
	0X00000005U,
	};

Configure the DMA channels (in 'DmaChannelConfig' container) indicated below (variant-aware. Variant name is 'Diesel'): • 3 Dma channels (Channel 0, 2, 4) to Core0 • 2 Dma channels (Channel 6, 10) to Core 1 Disable DmaPatternConfig container	const Dma_ConfigType Dma_Config_Diesel= { { /* Channel Configuration root for Core 0*/ /* This is the number of resources:3 */ &Dma_ChConfigRootCore0, /* Channel Configuration root

```
for Core 1*/
    /* This is the number of
resources:2 */
    &Dma_ChConfigRootCore1,
    /* Channel Configuration root
for Core 2*/
    /* This is the number of
resources:0 */
    NULL_PTR,
    /* Channel Configuration root
for Core 3*/
    /* This is the number of
resources:0 */
    NULL_PTR,
    /* Channel Configuration root
for Core 4*/
    /* This is the number of
resources:0 */
    NULL_PTR,
    /* Channel Configuration root
for Core 5*/
    /* This is the number of
resources:0 */
    NULL_PTR,
  },



  /* MoveEngine Error config for
ME0 and ME1 */
  {0X04030000U,0X04030000U},
  /* Access partition configuration
*/
  {
    /* Bus master configuration */

{0XFFFFFFFFU,0XFFFFFFFFU,0XFFFFFFFF
U,0XFFFFFFFFU},
    /* Resource partition
configuration */
    {  0X01U,  0X01U,  0X01U,
0X01U}
  },
  /* { Channel Position Index,
Channel Core Map } - The mapping
data of channels */
```

```
  {
    {0, 0 }, /* Channel 0 */
    {255, 255 }, /* Channel 1 */
    {1, 0 }, /* Channel 2 */
    {255, 255 }, /* Channel 3 */
    {2, 0 }, /* Channel 4 */
    {255, 255 }, /* Channel 5 */
    {0, 1 }, /* Channel 6 */
    {255, 255 }, /* Channel 7 */
    {255, 255 }, /* Channel 8 */
    {255, 255 }, /* Channel 9 */
    {1, 1 }, /* Channel 10 */
    {255, 255 }, /* Channel 11 */
    {255, 255 }, /* Channel 12 */
    .
    .
    .
    {255, 255 }, /* Channel 126 */
    {255, 255 }, /* Channel 127 */
  },
  {
    /* Bit map of the channels with
TRL enabled */
    0x0U,
    0x0U,
    0x0U,
    0x0U,
  },
  /* Total number of DMA channels
*/
  0X00000005U,
};
```

1.2.1.1 Member: DmaCoreConfigPtr[MCAL_NO_OF_CORES]

Table 28 DmaCoreConfigPtr[MCAL_NO_OF_CORES]

Name	DmaCoreConfigPtr[MCAL_NO_OF_CORES]
Type	Dma_CoreSpecificChConfigType *
Description	Array of core-specific configuration. The number of elements in this array is determined by the parameter ‘MCAL_NO_OF_CORES’, which indicates the number of CPU cores

	available in the hardware.
Verification method	The generated structure member is present in the Dma_Config[_<variant>] structure. If a Core<x> is allocated at least one channel (in ResourceM/ResourceMMcalConfig/ResourceMMcalConfig_0/ResourceMMcalCore container), then the element <x> will be generated as a pointer to structure Dma_CoreSpecificChConfigType (&Dma_ChConfigRootCore<x>) else 'NULL_PTR' is generated. (x in range 0 to 5).

Example(s)	**Action**	**Generated output**

	All the DMA channels are allocated to Core 0	`{`
		`  &Dma_ChConfigRootCore0,`
		`  NULL_PTR,`
		`  NULL_PTR,`
		`  NULL_PTR,`
		`  NULL_PTR,`
		`  NULL_PTR`
		`}`

	All the DMA channels are split between all cores except Core 0.	`{`
		`  NULL_PTR,`
		`  &Dma_ChConfigRootCore1,`
		`  &Dma_ChConfigRootCore2,`
		`  &Dma_ChConfigRootCore3,`
		`  &Dma_ChConfigRootCore4,`
		`  &Dma_ChConfigRootCore5`
		`}`

1.2.1.2 Member: DmaMovEngErr [2]

Table 29 DmaMovEngErr [2]

Name	DmaMovEngErr [2]
Type	uint32
Description	Array of two elements holding the configurations of different errors which can be triggered from any of the two move engines.
Verification method	This generated structure member contains the errors to be configured for the move engines. The following 3 bit positions are used to indicate the different errors: • DmaMESourceErrorInterrupt – Bit 16 • DmaMEDestinationErrorInterrupt – Bit 17 • DmaMELinkedListErrorInterrupt – Bit 26 • Other bits are unused and kept as zero.

Example(s)	**Action**	**Generated output**
	Disable the following errors of the move engine, in the 'DmaMoveEngineConfig' container:	`{ 0X00000000U,0X00000000U }`

	• DmaMESourceErrorInterrupt • DmaMEDestinationErrorInterrupt • DmaMELinkedListErrorInterrupt	
	Enable the following errors of the move engine, in the 'DmaMoveEngineConfig' container: • DmaMESourceErrorInterrupt • DmaMEDestinationErrorInterrupt • DmaMELinkedListErrorInterrupt	`{ 0X04030000U,0X04030000U }`

1.2.1.3 Member: DmaHwResourcePartition

Table 30 DmaHwResourcePartition

Name	DmaHwResourcePartition	
Type	Dma_ResourcePartitionConfigType	
Description	Structure holding the resource partition configuration of the DMA channels.	
Verification method	The generated structure member contains entries corresponding to the bus mode configuration (DmaResourcePartition/DmaResourcePartitionBusMode) and permitted bus master configuration (DmaResourcePartition/DmaPermittedBusMaster) of the resource partitions. The first parameter of 'DmaHwResourcePartition' holds the permitted bus masters, for each of the four Resource Partitions. The value is populated from parameter 'DmaResourcePartition/DmaPermittedBusMaster' as it is. This parameter has to be configured as mentioned in the Target Specifiction for the SFR ACCENr0. The second parameter of 'DmaHwResourcePartition' holds the bus mode configuration for each of the four Resource Partitions. The value is derived from the parameter 'DmaResourcePartition/DmaResourcePartitionBusMode' as follows: • 0x00U – if DMA_RP_USER_MODE is configured. • 0x01U – if DMA_RP_SUPERVISOR_MODE is configured. • Other values are unused for this parameter.	
Example(s)	**Action**	**Generated output**

	Configure the resource partitions as follows:
	• Resource Partition 0:
	Bus mode - User Mode
	Permitted bus master – 0xA5A5A5A5
	• All other resource partitions:
	Bus mode - Supervisor Mode
	Permitted bus master – 0xFFFFFFFF
		`{`
		`{0XA5A5A5A5U,`
		`0XFFFFFFFFU,`
		`0XFFFFFFFFU,`
		`0XFFFFFFFFU},`
		`{0X00U, 0X01U, 0X01U, 0X01U}`
		`}`

	Configure the resource partitions as follows:
	• Resource Partition 2:
	Bus mode - User Mode
	Permitted bus master – 0xA5A5A5A5
	• All other resource partitions:
	Bus mode - Supervisor Mode
	Permitted bus master – 0xFFFFFFFF
		`{`
		`{0XFFFFFFFFU,`
		`0XFFFFFFFFU,`
		`0XA5A5A5A5U,`
		`0XFFFFFFFFU},`
		`{0X01U, 0X01U, 0X00U, 0X01U}`
		`}`

1.2.1.4 Member: Dma_ChannelMaps[DMA_MAX_NUM_OF_CHANNELS]

Table 31 Dma_ChannelMaps[DMA_MAX_NUM_OF_CHANNELS]

Name	Dma_ChannelMaps[DMA_MAX_NUM_OF_CHANNELS]	
Type	Dma_ChannelMapType	
Description	This array holds the CPU core number and configuration index position of each DMA channel. The number of entries in this array corresponds to the number of DMA channels available in the hardware.	
Verification method	The array gets generated for each channel with the CPU core number (which is configured in ResourceM/ResourceMMcalConfig/ResourceMMcalCore/ResourceMAllocation container) and with the index to the configuration (which is configured in DmaChannelConfig container). If the channel is not configured, the value 255 is used for the parameter. The first parameter of 'Dma_ChannelMaps' holds the index location of the channel configuration in the 'Dma_kChConfigRoot_Core<x>' structure. The value is generated based on the sequence in which the channel configuration is added in the 'DmaChannelConfig' container, for a particular CPU core.The value can be from 0 to 127, depending on the number of channels allocated in the 'DmaChannelConfig' container. If the DMA channel is not configured, this value is kept as 255. The second parameter of 'Dma_ChannelMaps' holds the CPU core number to which the channel is allocated, in the 'ResourceM/ResourceMMcalConfig/ResourceMMcalCore/ResourceMAllocation' container. The value can be from 0 to the maximum number of CPU cores in the hardware. If the DMA channel is not configured, this value is kept as 255.	
Example(s)	**Action**	**Generated output**

	Configure the following DMA channels (in 'DmaChannelConfig' container) and allocate them to the CPU cores (in ResourceM/ResourceMMcalCo	`{`
		`{0, 0 }, /* Channel 0 */`
		`{255, 255 }, /* Channel 1 */`
		`{1, 0 }, /* Channel 2 */`

	nfig/ResourceMMcalCore/ResourceMAllocation container) as indicated: • Channel 0 – Core 0 • Channel 2 – Core 0 • Channel 4 – Core 0 • Channel 6 – Core 1 • Channel 10 – Core 1 • Channel 5 – Core 5 The order of the channel configuration in 'DmaChannelConfig' container has to be in the order mentioned above.	{255, 255 }, /* Channel 3 */ {2, 0 }, /* Channel 4 */ {0, 5 }, /* Channel 5 */ {0, 1 }, /* Channel 6 */ {255, 255 }, /* Channel 7 */ {255, 255 }, /* Channel 8 */ {255, 255 }, /* Channel 9 */ {1, 1 }, /* Channel 10 */ {255, 255 }, /* Channel 11 */ {255, 255 }, /* Channel 12 */ {255, 255 }, /* Channel 13 */ . . . . {255, 255 }, /* Channel 122 */ {255, 255 }, /* Channel 123 */ {255, 255 }, /* Channel 124 */ {255, 255 }, /* Channel 125 */ {255, 255 }, /* Channel 126 */ {255, 255 }, /* Channel 127 */ }

1.2.1.5 Member: DmaChTrlEnabled [DMA_MAX_NUM_OF_CHANNELS]

Table 32 DmaChTrlEnabled[DMA_MAX_TRLCHANNELS]

Name	DmaChTrlEnabled[DMA_MAX_TRLCHANNELS]	
Type	uint32	
Description	This array holds the bitfield to indicate the channels which have the Transaction Request LossTRL feature enabled.	
Verification method	The array gets generated with a bitfield which indicates the channels having the TRL feature enabled in the 'DmaTcsInterruptTransactionLoss' parameter in the 'DmaChannelConfig' container. The bit would be set if the feature is enabled for the channel. The bit 0 of the first array element stands for the channel 0 and the bit 31 of the fourth array element stands for the channel 127.	
Example(s)	**Action**	**Generated output**
	Enable the TRL feature for the following DMA channels (using 'DmaTcsInterruptTransactionLoss' parameter in 'DmaChannelConfig' container):	{ /* Bit map of the channels with TRL enabled */ 0x49U, 0x0U,

	• Channel 0 • Channel 3 • Channel 6 • Channel 121	0x0U, 0x2000000U, },

1.2.1.6 Member: DmaTotalChConfigured

Table 33 DmaTotalChConfigured

Name	DmaTotalChConfigured	
Type	uint8	
Description	Indicates the total number of DMA channels configured for the driver.	
Verification method	This value holds the total number of DMA channels configured across the cores in the DmaChannelConfig container.	
Example(s)	**Action**	**Generated output**
	Configure the following DMA channels (in 'DmaChannelConfig' container) and allocate them to the CPU cores (in ResourceM/ResourceMMcalConfig/ResourceMMcalCore/ResourceMAllocation container) as indicated: • Channel 0 – Core 0 • Channel 2 – Core 0 • Channel 4 – Not allocated • Channel 6 – Core 1 • Channel 10 – Not allocated • Channel 5 – Core 5	0X00000006U

1.2.2 Structure: Dma_ChConfigRootCore<x>

Table 34 Dma_ChConfigRootCore<x>

Name	Dma_ChConfigRootCore<x>	
Type	Dma_CoreSpecificChConfigType	
Description	Configuration structure of DMA driver for Core <x> (x ranges from 0 to 5).	
Verification method	The generated file has this structure if at least one channel is assigned to Core <x>, as in the ResourceM/ResourceMMcalConfig/ResourceMMcalCore/ResourceMAllocation container.	
Example(s)	**Action**	**Generated output**
	Configure 3 DMA channels to Core0 in 'DmaChannelConfig' container.	static const Dma_CoreSpecificChConfigType Dma_ChConfigRootCore0=

		{
		&Dma_kChConfigRoot_Core0[0],
		3
		};

1.2.2.1 Member: DmaChConfigPtr

Table 35 DmaChConfigPtr

Name	DmaChConfigPtr	
Type	Dma_ChConfigType*	
Description	Pointer to the base of array which stores the data of each channel configured to Core<x>.	
Verification method	The structure member is generated with base address of array which stores the channel data of Core <x>, as configured in the ResourceM/ResourceMMcalConfig/ResourceMMcalCore/ResourceMAllocation container.	
Example(s)	**Action**	**Generated output**
	Configure 1 DMA channel (in 'DmaChannelConfig' container) and assign it to Core 1 (in ResourceM/ResourceMMcalConfig/ResourceMMcalCore/ResourceMAllocation container).	& Dma_kChConfigRoot_Core1[0]
	Configure no DMA channels to Core 3 (in ResourceM/ResourceMMcalConfig/ResourceMMcalCore/ResourceMAllocation container).	NULL_PTR

1.2.2.2 Member: DmaNumberofChConfiguredPerCore

Table 36 DmaNumberofChConfiguredPerCore

Name	DmaNumberofChConfiguredPerCore	
Type	uint8	
Description	Indicates the total number of channels assigned to Core <x>.	
Verification method	The structure member is generated as total number of channels allocated to CORE<x>. *Note: Channels not assigned to any core are assigned to master core (ResourceMMasterCore).*	
Example(s)	**Action**	**Generated output**
	• Configure 4 DMA channels (in	3

	'DmaChannelConfig' container). • Peform the channel allocation in ResourceM/ResourceMMcalConfig /ResourceMMcalCore/ResourceM Allocation container: Core 0 – 3 channels Core 1 – 1 channel • Set the master core as CORE0, in the parameter ResourceM/ResourceMMcalConfig /ResourceMMasterCore. Output is shown for Core 0	
	• Configure 14 DMA channels (in 'DmaChannelConfig' container). • Allocate 3 channels to Core 1 in ResourceM/ResourceMMcalConfig /ResourceMMcalCore/ResourceM Allocation container. • Do not allocate the remaining 11 channels to any core. • Set the master core as CORE0, in the parameter ResourceM/ResourceMMcalConfig /ResourceMMasterCore. Output is shown for Core 0	`11`

1.2.3 Structure: Dma_kChConfigRoot_Core<x>

Table 37 Dma_kChConfigRoot_Core<x>

Name	Dma_kChConfigRoot_Core<x>	
Type	Dma_ChConfigType	
Description	Configuration structure of DMA driver holding the configuration of a particular DMA channel.	
Verification method	The generated file has this structure if at least one channel configured in the DmaChannelConfig container is assigned to Core <x> in the ResourceM/ResourceMMcalConfig/ResourceMMcalCore/ResourceMAllocation container.	
Example(s)	**Action**	**Generated output**

	• Configure DMA channel 0, DMA channel 2 and DMA channel 4 (in 'DmaChannelConfig' container).
	• Allocate the 3 channels to Core 0 in ResourceM/ResourceMMcalConf
		`static const Dma_ChConfigType Dma_kChConfigRoot_Core0[]=`
		`{`
		`  {`
		
		`&Dma_kChannel_0_TcsConfigRoot[0U]`

	ig/ResourceMMcalCore/ResourceMAllocation container.	,
		NULL_PTR,
		0x00000000U,
		(uint8)0U,
		(uint8)0x0U,
		NULL_PTR
		},
		{
		
		&Dma_kChannel_2_TcsConfigRoot[0U]
		,
		NULL_PTR,
		0x00000000U,
		(uint8)2U,
		(uint8)0x0U,
		NULL_PTR
		},
		{
		
		&Dma_kChannel_4_TcsConfigRoot[0U]
		,
		NULL_PTR,
		0x00000000U,
		(uint8)4U,
		(uint8)0x0U,
		NULL_PTR
		},
		};

1.2.3.1 Member: DmaChTCSPtr

Table 38 DmaChTCSPtr

Name	DmaChTCSPtr	
Type	Dma_TransactionCtrlSetType	
Description	Pointer to the Transaction Control Sets configured for the channel.	
Verification method	The structure member is generated with the pointer to the Transaction Control Sets of the channel in the DmaChannelConfig/DmaChannelTransactionSet container.	
Example(s)	**Action**	**Generated output**
	• Configure DMA channel 0 in the DmaChannelConfig container. • Configure a TCS for the DMA channel 0 in	&Dma_kChannel_0_TcsConfigRoot[0U]

	DmaChannelConfig/DmaChannelTransactionSet container	
	• Configure DMA channel 4 in the DmaChannelConfig container. • Configure a TCS for the DMA channel 4 in DmaChannelConfig/DmaChannelTransactionSet container	`&Dma_kChannel_4_TcsConfigRoot[0U]`

1.2.3.2 Member: DmaChNotifPtr

Table 39 DmaChNotifPtr

Name	DmaChNotifPtr	
Type	Dma_ChannelNotificationPtrType	
Description	The notification function which would be invoked for the DMA Channel Interrupt Service Request.	
Verification method	The structure member is generated with the function pointer configured for the notification function in the parameter DmaChannelConfig/DmaChannelNotification	
Example(s)	**Action**	**Generated output**
	Configure a function pointer 'Dma_Notifiation_Isr', in DmaChannelConfig/DmaChannelNotification parmeter.	`Dma_Notifiation_Isr`
	Configure 'NULL_PTR' as the function pointer, in DmaChannelConfig/DmaChannelNotification parmeter.	`NULL_PTR`

1.2.3.3 Member: DmaTsrConfig

Table 40 DmaTsrConfig

Name	DmaTsrConfig	
Type	uint32	
Description	The parameter which holds the TSR register configuration for the DMA channel.	
Verification method	The bit position 4 of this data indicates the Transaction Loss interrupt enable/disable status as configured in the DmaChannelConfig/DmaTcsInterruptTransactionLoss parameter. Other bits are unused.	
Example(s)	**Action**	**Generated output**
	Enable the Transaction Loss interrupt, by setting the DmaChannelConfig/DmaTcsInterruptTransactionLoss parameter	`0x00000010U`
	Disable the Transaction Loss interrupt, by clearing the DmaChannelConfig/DmaTcsInterruptTransactionLoss parameter	`0x00000000U`

1.2.3.4 Member: DmaChNumber

Table 41 DmaChNumber

Name	DmaChNumber	
Type	uint8	
Description	The channel number of the DMA channel.	
Verification method	The structure member indicates the channel number of the DMA channel configured in the parameter DmaChannelConfig/DmaChannelId.	
Example(s)	**Action**	**Generated output**
	Configure a DMA channel in the DmaChannelConfig container and set the channel ID as 2 in the DmaChannelConfig/DmaChannelId parameter.	`(uint8)2U`
	Configure a DMA channel in the DmaChannelConfig container and set the channel ID as 127 in the DmaChannelConfig/DmaChannelId parameter.	`(uint8)127U`

1.2.3.5 Member: DmaChHwPartitionConfig

Table 42 DmaChHwPartitionConfig

Name	DmaChHwPartitionConfig	
Type	uint8	
Description	The access partition configured for the DMA channel.	
Verification method	The structure member is generated with the access partition configured in the parameter DmaChannelConfig/DmaChannelAssignedPartition.	
Example(s)	**Action**	**Generated output**
	Set the Assigned partition as 2, in the ‘DmaChannelConfig/DmaChannelAssignedPartition’ parameter	`(uint8)0x2U`

1.2.3.6 Member: DmaMEErrorNotifPtr

Table 43 DmaMEErrorNotifPtr

Name	DmaMEErrorNotifPtr	
Type	Dma_MoveEngineErrorNotificationPtrType	
Description	The notification function which would be invoked for the DMA Resource Partition Error Interrupt Service Request.	
Verification method	The structure member is generated with the function pointer configured for the notification function in the parameter DmaResourcePartition/DmaMoveEngineErrorNotifRoutine.	
Example(s)	**Action**	**Generated output**

	Configure a function pointer 'Dma_Error_Isr' in the 'DmaResourcePartition/DmaMoveEngineErrorNotifRoutine' parameter.	`Dma_Error_Isr`
	Configure 'NULL_PTR' as the function pointer, in the 'DmaResourcePartition/DmaMoveEngineErrorNotifRoutine' parameter.	`NULL_PTR`

1.2.4 Structure: Dma_kChannel_<x>_TcsConfigRoot

Table 44 Dma_kChannel_<x>_TcsConfigRoot

Name	Dma_kChannel_<x>_TcsConfigRoot	
Type	Dma_TransactionCtrlSetType	
Description	Configuration structure holding the Transaction Control Set of the channel number <x>. (<x> can range from 0 to the number of channels available in the hardware).	
Verification method	The generated file has this structure with the Transaction Control Settings configured in the DmaChannelConfig/DmaChannelTransactionSet container.	
Example(s)	**Action**	**Generated output**
	Configure the DMA channel 5, with the following data (in the DmaChannelConfig/DmaChannelTransactionSet container): • DmaTcsIndex = 0 • DmaTcsSourceAddress = 0x12345678 • DmaTcsDestinationAddress = 0x23456789 • DmaTcsDoubleBuffer = 0 • DmaTcsMoveLength = DMA_WIDTH_8BITS • DmaTcsTransferLength = DMA_MOVES_1 • DmaTcsTransactionLength = 5 • DmaTcsCircularBufferSourceEnable = False • DmaTcsCircularBufferDestinationEnable = False • DmaTcsCircularBufferSourceLength = DMA_CIRCULAR_BUFFER_LENGTH_1BYTE • DmaTcsCircularBufferDestinationLength = DMA_CIRCULAR_BUFFER_LENGT	`static const Dma_TransactionCtrlSetType Dma_kChannel_5_TcsConfigRoot[]= { /* Dma Channel 5 TCS 0 */ { /* Configuration for DMA register DMA_RDCRCRz */ 0x00000000U, /* Configuration for DMA register DMA_SDCRCRz */ 0x00000000U, /* Configuration for DMA source address register DMA_SADRz */ /* MISRA2012_RULE_11_6_JUSTIFICATION: Shadow SFR access */ /* MISRA2012_RULE_11_4_JUSTIFICATION: Shadow SFR access */ (uint32 *)0x12345678U, /* Configuration for DMA destination address register DMA_DADRz */ /*`

H_1BYTE
• DmaTcsSourceAddressModificationFactor = DMA_FACTOR_1
• DmaTcsDestinationAddressModificationFactor = DMA_FACTOR_1
• DmaTcsAppendTimeStamp = False
• DmaTcsSourceAddressMovement = DMA_INCREASING
• DmaTcsDestinationAddressMovement = DMA_INCREASING
• DmaTcsShadowRegisterConfiguration = DMA_LINKED_LIST
• DmaNextTcsIndex = 0
• DmaTcsTriggerFrequency = DMA_TRANSFER_PER_TRIGGER
• DmaTcsHardwareTrigger = DMA_NO_HARDWARE_TRIGGER
• DmaTcsDaisyChaining = False
• DmaTcsInterruptSourceAddressWrap = False
• DmaTcsInterruptDestinationAddressWrap = False
• DmaTcsDataTransferInterrupt = False
• DmaTcsInterruptDataTransfer = DMA_INTERRUPT_PER_TRANSACTION
• DmaTcsInterruptDataTransferThreshold = 0
• DmaTcsSwapDataCRCByteOrder = False
• DmaTcsAutoStartEnable = False
• DmaTcsReferenceAddressCrc = 0
• DmaTcsReferenceDataCrc = 0
	`MISRA2012_RULE_11_6_JUSTIFICATION: Shadow SFR access */`
	`/* MISRA2012_RULE_11_4_JUSTIFICATION: Shadow SFR access */`
	`(uint32 *)0x23456789U,`
	`/* Configuration for DMA Channel Address and Interrupt Control`
	`* Register DMA_ADICRz */`
	`0x000c0088U,`
	`/* Configuration for DMA Channel Configuration Register DMA_CHCFGRz */`
	`0x00000005U,`
	`/* Configuration for DMA Channel Shadow Address Register DMA_SHADRz */`
	`/* MISRA2012_RULE_11_6_JUSTIFICATION: Shadow SFR access */`
	`/* MISRA2012_RULE_11_4_JUSTIFICATION: Shadow SFR access */`
	`(const struct Dma_TransactionCtrlSetType*)&Dma_kChannel_5_TcsConfigRoot[0],`
	`/* Configuration for DMA Channel Control and Status Register DMA_CHCSRz */`
	`0x00000000U`
	`}`
	`};`

1.2.4.1 Member: DmaReadDataCrc

Table 45 DmaReadDataCrc

Name	DmaReadDataCrc
Type	uint32
Description	The data CRC value configured for the DMA channel.
Verification method	The structure member is generated with the Data CRC configured in the parameter DmaChannelConfig/DmaChannelTransactionSet/DmaTcsReferenceDataCrc.

	Note: This parameter would be generated only if the parameter 'DmaChannelConfig/DmaChannelTransactionSet/DmaTcsShadowRegisterConfiguration' is configured as either: 'DMA_LINKED_LIST' or 'DMA_ACCUMULATED_LINKED_LIST' or 'DMA_SAFE_LINKED_LIST' or 'DMA_CONDITIONAL_LINKED_LIST', for any of the channels configured in the ' DmaChannelConfig' container.

Example(s)	Action	Generated output
	Configure the DMA channel 5 with data CRC as '0xa5a5a5a5, in the parameter 'DmaChannelConfig/DmaChannelTransactionSet/DmaTcsReferenceDataCrc'.	`0xa5a5a5a5U`

1.2.4.2 Member: DmaSourceDestAddressCrc

Table 46 DmaSourceDestAddressCrc

Name	DmaSourceDestAddressCrc
Type	uint32
Description	The address CRC value configured for the DMA channel.
Verification method	The structure member is generated with the address CRC configured in the parameter DmaChannelConfig/DmaChannelTransactionSet/DmaTcsReferenceAddressCrc. *Note: This parameter would be generated only if the parameter 'DmaChannelConfig/DmaChannelTransactionSet/DmaTcsShadowRegisterConfiguration' is configured as either: 'DMA_LINKED_LIST' or 'DMA_ACCUMULATED_LINKED_LIST' or 'DMA_SAFE_LINKED_LIST' or 'DMA_CONDITIONAL_LINKED_LIST', for any of the channels configured in the ' DmaChannelConfig' container.*

Example(s)	Action	Generated output
	Configure the DMA channel 5 with address CRC as '0x5a5a5a5a', in the parameter 'DmaChannelConfig/DmaChannelTransactionSet/ DmaTcsReferenceAddressCrc'.	`0x5a5a5a5aU`

1.2.4.3 Member: DmaSourceAddress

Table 47 DmaSourceAddress

Name	DmaSourceAddress
Type	uint32*
Description	The parameter holds the source address from where the data needs to be copied from.

Verification method	The structure member is generated with the source address configured in the parameter 'DmaChannelConfig/DmaChannelTransactionSet/DmaTcsSourceAddress'. However, if variable names are used instead of address values, the header file which contains the extern declaration of the variable name should be provided in the dependent parameter 'DmaChannelConfig/DmaChannelTransactionSet/DmaUserHeaderFileWithExternDeclarations'.	
Example(s)	**Action**	**Generated output**
	Configure the DMA channel 5 with source address as '0x12345678', in the parameter 'DmaChannelConfig/DmaChannelTransactionSet/DmaTcsSourceAddress'.	`(uint32 *)0x12345678U`

	Configure the DMA channel 6, with the following data (in the DmaChannelConfig/DmaChannelTransactionSet container), for the TCS 0:
	• DmaTcsSourceAddress = &SourceAddressVar
	Provide the header file name, which contains the declaration of the variable specified for the 'DmaTcsSourceAddress' parameter:
	• DmaUserHeaderFileWithExternDeclarations = VariableHeaderFile.h
		`#include "VariableHeaderFile.h"`
		.
		.
		.
		`static const Dma_TransactionCtrlSetType Dma_kChannel_6_TcsConfigRoot[]=`
		`{`
		`/* Dma Channel 6 TCS 0 */`
		`{`
		.
		.
		.
		`(uint32 *)&SourceAddressVar,`
		.
		.
		`}`
		`};`

	Configure the DMA channel 6, with the following data (in the DmaChannelConfig/DmaChannelTransactionSet container), for the TCS 0:
	• DmaTcsSourceAddress = &SourceAddressVar
	Provide the header file name, which contains the declaration of the variable specified for the 'DmaTcsSourceAddress' parameter:
	• DmaUserHeaderFileWithExternDeclarations = VariableHeaderFile.h
		`#include "VariableHeaderFile.h"`
		`#include "VariableHeaderFile2.h"`
		.
		.
		.
		`static const Dma_TransactionCtrlSetType Dma_kChannel_6_TcsConfigRoot[]=`

	Configure the TCS1 also, in the in the 'DmaChannelConfig/DmaChannelTransactionSet' container as follows:
	• DmaTcsSourceAddress = &SourceAddressVar2
	Provide the header file name, which contains the declaration of the variable specified for the 'DmaTcsSourceAddress' parameter:
	• DmaUserHeaderFileWithExternDeclarations = VariableHeaderFile2.h
		{
		/* Dma Channel 6 TCS 0 */
		{
		.
		.
		.
		(uint32 *)&SourceAddressVar,
		.
		.
		},
		/* Dma Channel 6 TCS 1 */
		{
		.
		.
		.
		(uint32 *)&SourceAddressVar2,
		.
		.
		}
		};

1.2.4.4 Member: DmaDestinationAddress

Table 48 DmaDestinationAddress

Name	DmaDestinationAddress	
Type	uint32*	
Description	The parameter holds the destination address to which the data needs to be copied to.	
Verification method	The structure member is generated with the destination address configured in the parameter DmaChannelConfig/DmaChannelTransactionSet/DmaTcsDestinationAddress. However, if variable names are used instead of address values, the header file which contains the extern declaration of the variable name should be provided in the dependent parameter 'DmaChannelConfig/DmaChannelTransactionSet/DmaUserHeaderFileWithExternDeclarations'.	
Example(s)	**Action**	**Generated output**
	Configure the DMA channel 5 with source address as '0x23456789', in 'DmaChannelConfig/DmaChannelTransactionSet/DmaTcsDestinationAddress' parameter.	(uint32 *)0x23456789U

Configure the DMA channel 6, with the following data (in the DmaChannelConfig/DmaChannelTransactionSet container), for the TCS 0:
• DmaTcsDestinationAddress = &DestAddressVar
Provide the header file name, which contains the declaration of the variable specified for the 'DmaTcsDestinationAddress' parameter:
DmaUserHeaderFileWithExternDeclarations = VariableHeaderFile.h
	`#include "VariableHeaderFile.h"`
	.
	.
	.
	`static const Dma_TransactionCtrlSetType Dma_kChannel_6_TcsConfigRoot[ ]=`
	`{`
	`/* Dma Channel 6 TCS 0 */`
	`{`
	.
	.
	.
	`(uint32 *)&DestAddressVar,`
	.
	.
	`}`
	`};`

Configure the DMA channel 6, with the following data (in the DmaChannelConfig/DmaChannelTransactionSet container), for the TCS 0:
• DmaTcsDestinationAddress = &DestAddressVar
Provide the header file name, which contains the declaration of the variable specified for the 'DmaTcsDestinationAddress' parameter:
• DmaUserHeaderFileWithExternDeclarations = VariableHeaderFile.h
 	
Configure the TCS1 also, in the in the 'DmaChannelConfig/DmaChannelTransactionSet' container as follows:
• DmaTcsDestinationAddress = &DestAddressVar2
Provide the header file name, which contains the declaration of the variable specified for the 'DmaTcsDestinationAddress' parameter:
• DmaUserHeaderFileWithExternDeclara
	`#include "VariableHeaderFile.h"`
	`#include "VariableHeaderFile2.h"`
	.
	.
	.
	`static const Dma_TransactionCtrlSetType Dma_kChannel_6_TcsConfigRoot[ ]=`
	`{`
	`/* Dma Channel 6 TCS 0 */`
	`{`
	.
	.
	.
	`(uint32 *)&DestAddressVar,`
	.
	.

	tions = VariableHeaderFile2.h	},
		/* Dma Channel 6 TCS 1 */
		{
		.
		.
		.
		(uint32 *)&DestAddressVar2,
		.
		.
		}
		};

1.2.4.5 Member: DmaAddressInterruptControl

Table 49 DmaAddressInterruptControl

Name	DmaAddressInterruptControl
Type	uint32
Description	This parameter holds the register value to be configured for the ADICR register of the DMA channel.
Verification method	The structure member indicates the ADICR register value from the configuration settings kept in the DmaChannelConfig/DmaChannelTransactionSet container. The parameters used, and the method used to derive the values for ADICR register from these parameters are illustrated below: • Bit 0-2: DmaTcsSourceAddressModificationFactor. This is calculated as follows: 000 - DMA_FACTOR_1 001 - DMA_FACTOR_2 010 - DMA_FACTOR_4 011 - DMA_FACTOR_8 100 - DMA_FACTOR_16 101 - DMA_FACTOR_32 110 - DMA_FACTOR_64 111 - DMA_FACTOR_128 • Bit 3: DmaTcsSourceAddressMovement. This is calculated as follows: 0 - DMA_DECREASING 1 - DMA_INCREASING • Bit 4 -6: DmaTcsDestinationAddressModificationFactor. This is calculated as follows: 000 - DMA_FACTOR_1 001 - DMA_FACTOR_2 010 - DMA_FACTOR_4 011 - DMA_FACTOR_8 100 - DMA_FACTOR_16 101 - DMA_FACTOR_32

	110 - DMA_FACTOR_64 111 - DMA_FACTOR_128 • Bit 7: DmaTcsDestinationAddressMovement. This is calculated as follows: 0 - DMA_DECREASING 1 - DMA_INCREASING • Bit 8-11: DmaTcsCircularBufferSourceLength. This is calculated as follows: 0000 - DMA_CIRCULAR_BUFFER_LENGTH_1BYTE 0001 - DMA_CIRCULAR_BUFFER_LENGTH_2BYTE 0010 - DMA_CIRCULAR_BUFFER_LENGTH_4BYTE 0011 - DMA_CIRCULAR_BUFFER_LENGTH_8BYTE 0100 - DMA_CIRCULAR_BUFFER_LENGTH_16BYTE 0101 - DMA_CIRCULAR_BUFFER_LENGTH_32BYTE 0110 - DMA_CIRCULAR_BUFFER_LENGTH_64BYTE 0111 - DMA_CIRCULAR_BUFFER_LENGTH_128BYTE 1000 - DMA_CIRCULAR_BUFFER_LENGTH_256BYTE 1001 - DMA_CIRCULAR_BUFFER_LENGTH_512BYTE 1010 - DMA_CIRCULAR_BUFFER_LENGTH_1024BYTE 1011 - DMA_CIRCULAR_BUFFER_LENGTH_2048BYTE 1100 - DMA_CIRCULAR_BUFFER_LENGTH_4096BYTE 1101 - DMA_CIRCULAR_BUFFER_LENGTH_8192BYTE 1110 - DMA_CIRCULAR_BUFFER_LENGTH_16384BYTE 1111 - DMA_CIRCULAR_BUFFER_LENGTH_32768BYTE • Bit 12-15: DmaTcsCircularBufferDestinationLength. This is calculated as follows: 0000 - DMA_CIRCULAR_BUFFER_LENGTH_1BYTE 0001 - DMA_CIRCULAR_BUFFER_LENGTH_2BYTE 0010 - DMA_CIRCULAR_BUFFER_LENGTH_4BYTE 0011 - DMA_CIRCULAR_BUFFER_LENGTH_8BYTE 0100 - DMA_CIRCULAR_BUFFER_LENGTH_16BYTE 0101 - DMA_CIRCULAR_BUFFER_LENGTH_32BYTE 0110 - DMA_CIRCULAR_BUFFER_LENGTH_64BYTE 0111 - DMA_CIRCULAR_BUFFER_LENGTH_128BYTE 1000 - DMA_CIRCULAR_BUFFER_LENGTH_256BYTE 1001 - DMA_CIRCULAR_BUFFER_LENGTH_512BYTE 1010 - DMA_CIRCULAR_BUFFER_LENGTH_1024BYTE 1011 - DMA_CIRCULAR_BUFFER_LENGTH_2048BYTE 1100 - DMA_CIRCULAR_BUFFER_LENGTH_4096BYTE 1101 - DMA_CIRCULAR_BUFFER_LENGTH_8192BYTE 1110 - DMA_CIRCULAR_BUFFER_LENGTH_16384BYTE 1111 - DMA_CIRCULAR_BUFFER_LENGTH_32768BYTE • Bit 16-19: DmaTcsShadowRegisterConfiguration. This is calculated as follows: 0000 - DMA_SHADOWING_DISABLED 0001 - DMA_SOURCE_ADDRESS_BUFFERING_RO 0010 - DMA_DEST_ADDRESS_BUFFERING_RO 0011 - Reserved value, not used. 0100 - Reserved value, not used. 0101 - DMA_SOURCE_ADDRESS_BUFFERING_RW 0110 - DMA_DEST_ADDRESS_BUFFERING_RW 0111 - Reserved value, not used.

	1000 - DMA_SOURCE_DOUBLE_BUFFERING_SW_SWITCH 1001 - DMA_SOURCE_DOUBLE_BUFFERING_HW_SW_SWITCH 1010 - DMA_DEST_DOUBLE_BUFFERING_SW_SWITCH 1011 - DMA_DEST_DOUBLE_BUFFERING_HW_SW_SWITCH 1100 - DMA_LINKED_LIST 1101 - DMA_ACCUMULATED_LINKED_LIST 1110 - DMA_SAFE_LINKED_LIST 1111 - DMA_CONDITIONAL_LINKED_LIST • Bit 20: DmaTcsCircularBufferSourceEnable. This is calculated as follows: 0 – If 'DmaTcsCircularBufferSourceEnable' is disabled. 1 – If 'DmaTcsCircularBufferSourceEnable' is enabled. • Bit 21: DmaTcsCircularBufferDestinationEnable. This is calculated as follows: 0 – If 'DmaTcsCircularBufferDestinationEnable' is disabled. 1 – If 'DmaTcsCircularBufferDestinationEnable' is enabled. • Bit 22: DmaTcsAppendTimeStamp. This is calculated as follows: 0 – If 'DmaTcsAppendTimeStamp' is disabled. 1 – If 'DmaTcsAppendTimeStamp' is enabled. • Bit 23: Reserved, kept as zero. • Bit 24: DmaTcsInterruptSourceAddressWrap. This is calculated as follows: 0 – If 'DmaTcsInterruptSourceAddressWrap' is disabled. 1 – If 'DmaTcsInterruptSourceAddressWrap' is enabled. • Bit 25: DmaTcsInterruptDestinationAddressWrap. This is calculated as follows: 0 – If 'DmaTcsInterruptDestinationAddressWrap' is disabled. 1 – If 'DmaTcsInterruptDestinationAddressWrap' is enabled. • Bit 26-27: This parameter is set as follows: 00 - When 'DmaTcsDataTransferInterrupt' is false. 10 - Unused value. 10 - When 'DmaTcsDataTransferInterrupt' is true and 'DmaTcsInterruptDataTransfer' = 'DMA_INTERRUPT_AFTER_THRESHOLD' or 'DMA_INTERRUPT_PER_TRANSACTION' 11 - When 'DmaTcsDataTransferInterrupt' is true and 'DmaTcsInterruptDataTransfer' = 'DMA_INTERRUPT_PER_TRANSFER' • Bit 28-31: DmaTcsInterruptDataTransferThreshold. The value set for this parameter is directly copied to this register field.	
Example(s)	**Action**	**Generated output**
	Configure the DMA channel 5, with the following data (in the DmaChannelConfig/DmaChannelTransactionSet container): • DmaTcsTransactionLength = 5 • DmaTcsShadowRegisterConfiguration = DMA_LINKED_LIST • DmaNextTcsIndex = 0 • DmaTcsSourceAddressModificationFactor = DMA_FACTOR_1	`0x000c0088U,`

	• DmaTcsDestinationAddressModificationFactor = DMA_FACTOR_1 • DmaTcsSourceAddressMovement = DMA_INCREASING • DmaTcsDestinationAddressMovement = DMA_INCREASING • DmaTcsCircularBufferSourceEnable = False • DmaTcsCircularBufferDestinationEnable = False • DmaTcsAppendTimeStamp = False • DmaTcsCircularBufferSourceLength = DMA_CIRCULAR_BUFFER_LENGTH_1BYTE • DmaTcsCircularBufferDestinationLength = DMA_CIRCULAR_BUFFER_LENGTH_1BYTE • DmaTcsInterruptDestinationAddressWrap = False • DmaTcsInterruptSourceAddressWrap = False • DmaTcsDataTransferInterrupt = False • DmaTcsInterruptDataTransferThreshold = 0	

1.2.4.6 Member: DmaChannelConfig

Table 50 DmaChannelConfig

Name	DmaChannelConfig
Type	uint32
Description	This parameter holds the register value to be configured for the CHCFGR register of the DMA channel.
Verification method	The structure member indicates the CHCFGR register value from the configuration settings kept in the DmaChannelConfig/DmaChannelTransactionSet container. The value for the CHCFGR is derived from the parameters as follows: • Bit 0-13: The value is directly copied from the parameter DmaTcsTransactionLength. • Bit 16-18: DmaTcsTransferLength 000 - DMA_MOVES_1 001 - DMA_MOVES_2 010 - DMA_MOVES_4 011 - DMA_MOVES_8 100 - DMA_MOVES_16 101 - DMA_MOVES_3 110 - DMA_MOVES_5 111 - DMA_MOVES_9 • Bit 19: DmaTcsTriggerFrequency 0 - If DMA_TRANSACTION_PER_TRIGGER is set as DMA_TRANSFER_PER_TRIGGER

	1 - If DMA_TRANSACTION_PER_TRIGGER is set as DMA_TRANSACTION_PER_TRIGGER • Bit 20: DmaTcsHardwareTrigger 0 - If DmaTcsHardwareTrigger is set as DMA_HARDWARE_TRIGGER_SINGLE_MODE 1 - If DmaTcsHardwareTrigger is set as DMA_HARDWARE_TRIGGER_CONTINUOUS_MODE • Bit 21-23: DmaTcsMoveLength 000 - DMA_WIDTH_8BITS 001 - DMA_WIDTH_16BITS 010 - DMA_WIDTH_32BITS 011 - DMA_WIDTH_64BITS 100 - DMA_WIDTH_128BITS 101 - DMA_WIDTH_256BITS 110 - Reserved values, not used. 111 - Reserved values, not used. • Bit 24-26: Unused, kept as zero. • Bit 27: DmaTcsSwapDataCRCByteOrder 0 – If DmaTcsSwapDataCRCByteOrder is not set. 1 – If DmaTcsSwapDataCRCByteOrder is set. • Bit 28: DmaTcsDaisyChaining 0 – If DmaTcsDaisyChaining is not set. 1 – If DmaTcsDaisyChaining is set.	
Example(s)	**Action**	**Generated output**
	Configure the DMA channel 5, with the following data (in the DmaChannelConfig/DmaChannelTransactionSet container): • DmaTcsMoveLength = DMA_WIDTH_8BITS • DmaTcsTransferLength = DMA_MOVES_1 • DmaTcsTriggerFrequency = DMA_TRANSFER_PER_TRIGGER • DmaTcsTransactionLength = 5 • DmaTcsHardwareTrigger = DMA_NO_HARDWARE_TRIGGER • DmaTcsSwapDataCRCByteOrder = False • DmaTcsDaisyChaining = False • DmaTcsShadowRegisterConfiguration = DMA_LINKED_LIST • DmaNextTcsIndex = 0	`0x00000005U`

1.2.4.7 Member: DmaShadowAddress

Table 51 DmaShadowAddress

Name	DmaChNumber
Type	Dma_TransactionCtrlSetType*

Description	This parameter holds the pointer which points either to: • The next Transaction Control Set, if linked list feature is used. • The second buffer, if double buffer feature is used.
Verification method	The structure member gets generated with the pointer to the next Transaction Control Set (TCS) (if linked list feature is configured) or the second buffer address specified in the 'DmaTcsDoubleBuffer' parameter (if double buffer feature is configured). However, if variable names are used instead of address values, the header file which contains the extern declaration of the variable name should be provided in the dependent parameter 'DmaChannelConfig/DmaChannelTransactionSet/DmaUserHeaderFileWithExternDeclarations'. *Note: This parameter would be generated only if the parameter 'DmaChannelConfig/DmaChannelTransactionSet/DmaTcsShadowRegisterConfiguration' is configured as either: 'DMA_LINKED_LIST' or 'DMA_ACCUMULATED_LINKED_LIST' or 'DMA_SAFE_LINKED_LIST' or 'DMA_CONDITIONAL_LINKED_LIST'or 'DMA_SOURCE_DOUBLE_BUFFERING_SW_SWITCH' or 'DMA_SOURCE_DOUBLE_BUFFERING_HW_SW_SWITCH' or 'DMA_DEST_DOUBLE_BUFFERING_SW_SWITCH' or 'DMA_DEST_DOUBLE_BUFFERING_HW_SW_SWITCH', for any of the channels configured in the ' DmaChannelConfig' container.*

Example(s)	**Action**	**Generated output**
	Configure the max transaction sets for the DMA channels as: • DmaGeneral/DmaMaxTransactionSetPerChannel = 2 Configure the number of transaction sets for the DMA channel 5 as: • DmaChannelConfig/DmaChannelNumTransactionSet = 2 Add two TCSes for the DMA channel 5 in the container 'DmaChannelConfig/DmaChannelConfig_5/DmaChannelTransactionSet' Configure the DMA channel 5, with the following data (in the DmaChannelConfig/DmaChannelTransactionSet container), for the TCS 0:	`(const struct Dma_TransactionCtrlSetType*)&Dma_kChannel_5_TcsConfigRoot[1]`

• DmaTcsIndex = 0 • DmaTcsShadowRegisterConfiguration = DMA_LINKED_LIST • DmaNextTcsIndex = 1 For the DMA channel 5, configure the following data (in the DmaChannelConfig/DmaChannelTransactionSet container), for the TCS 1: • DmaTcsIndex = 1 • DmaTcsShadowRegisterConfiguration = DMA_SHADOWING_DISABLED Output is shown for the TCS 0, for the DMA channel.	

Configure the DMA channel 6, with the following data (in the DmaChannelConfig/DmaChannelTransactionSet container), for the TCS 0:
• DmaTcsShadowRegisterConfiguration = DMA_SOURCE_DOUBLE_BUFFERING_SW_SWITCH
• DmaTcsDoubleBuffer = &DoubleBufferVariable
Provide the header file name, which contains the declaration of the variable specified for the 'DmaTcsDoubleBuffer' parameter
• DmaUserHeaderFileWithExternDeclarations = VariableHeaderFile.h
	`#include "VariableHeaderFile.h"`
	.
	.
	`static const Dma_TransactionCtrlSetType Dma_kChannel_6_TcsConfigRoot[]=`
	`{`
	`/* Dma Channel 6 TCS 0 */`
	`{`
	.
	.
	.
	`(const struct Dma_TransactionCtrlSetType*)&DoubleBufferVariable,`
	.
	.
	`}`
	`};`

Configure the DMA channel 7, with the following data (in the DmaChannelConfig/DmaChannelTransactionSet container), for the TCS 0:
• DmaTcsShadowRegisterConfiguration = DMA_SOURCE_DOUBLE_BUFFERING_SW_SWITCH
• DmaTcsDoubleBuffer =
	`#include "VariableHeaderFile.h"`
	`#include "VariableHeaderFile2.h"`
	.
	.
	.
	`static const`

	&DoubleBufferVariable Provide the header file name, which contains the declaration of the variable specified for the 'DmaTcsDoubleBuffer' parameter DmaUserHeaderFileWithExternDeclarations = VariableHeaderFile.h Configure the DMA channel 8, with the following data (in the DmaChannelConfig/DmaChannelTransactionSet container), for the TCS 0: • DmaTcsShadowRegisterConfiguration = DMA_DEST_DOUBLE_BUFFERING_SW_SWITCH • DmaTcsDoubleBuffer = &DoubleBufferVariable2 Provide the header file name, which contains the declaration of the variable specified for the 'DmaTcsDoubleBuffer' parameter DmaUserHeaderFileWithExternDeclarations = VariableHeaderFile2.h	`Dma_TransactionCtrlSetType Dma_kChannel_7_TcsConfigRoot[]= { /* Dma Channel 7 TCS 0 */ { . . . (const struct Dma_TransactionCtrlSetType*)&DoubleBufferVariable, . . } }; static const Dma_TransactionCtrlSetType Dma_kChannel_8_TcsConfigRoot[]= { /* Dma Channel 8 TCS 0 */ { . . . (const struct Dma_TransactionCtrlSetType*)DoubleBufferVariable2, . . } };`

1.2.4.8 Member: DmaChControlStatus

Table 52 DmaChControlStatus

Name	DmaChControlStatus
Type	uint32
Description	This parameter holds the register value to be configured for the CHCSR register of the DMA channel.
Verification method	The structure member indicates the CHCSR register value from the channel autostart setting kept in the 'DmaChannelConfig/DmaChannelTransactionSet/DmaTcsAutoStartEnable'

	parameter. The bit 31 is set if the parameter DmaTcsAutoStartEnable is enabled, else it is kept as zero. The bits 0-31 are kept as zero. *Note: This parameter would be generated only if the parameter 'DmaChannelConfig/DmaChannelTransactionSet/DmaTcsShadowRegisterConfiguration' is configured as either: 'DMA_LINKED_LIST' or 'DMA_ACCUMULATED_LINKED_LIST' or 'DMA_SAFE_LINKED_LIST' or 'DMA_CONDITIONAL_LINKED_LIST', for any of the channels configured in the ' DmaChannelConfig' container.*

Example(s)	**Action**	**Generated output**
	Enable the autostart feature of the DMA channel 5 as: • DmaTcsAutoStartEnable = True	`0x80000000U`
	Disable the autostart feature of the DMA channel 5 as: • DmaTcsAutoStartEnable = False	`0x00000000U`

1.2.5 Function declaration: Dma_ChannelNotificationPtrType

Table 53 Dma_ChannelNotificationPtrType

Name	Dma_ChannelNotificationPtrType
Type	Dma_ChannelNotificationPtrType*
Description	The extern declaration of the user defined notification function which would be invoked for the DMA Channel Interrupt Service Request.
Verification method	The function configured in 'DmaChannelConfig/DmaChannelNotification' would be populated as a prototype with extern qualifier. *Note: This prototype would not be generated if the function is configured as NULL_PTR in 'DmaChannelConfig/DmaChannelNotification'.*

Example(s)	**Action**	**Generated output**
	• Configure the DMA channel 0 in 'DmaChannelConfig' container. • Configure the notification function as 'Dma_ChannelInterrupt'	`extern void Dma_ChannelInterrupt(uint8 Channel, uint32 Event);`

1.2.6 Function declaration: Dma_MoveEngineErrorNotificationPtrType

Table 54 Dma_MoveEngineErrorNotificationPtrType

Name	Dma_MoveEngineErrorNotificationPtrType
Type	Dma_MoveEngineErrorNotificationPtrType*

Description	The extern declaration of the user defined notification function which would be invoked for the DMA Resource Partition Error Interrupt Service Request.	
Verification method	The function configured in 'DmaResourcePartition/DmaMoveEngineErrorNotifRoutine' would be populated as a prototype with extern qualifier. *Note: This prototype would not be generated if the function is configured as NULL_PTR in 'DmaResourcePartition/DmaMoveEngineErrorNotifRoutine''*	
Example(s)	**Action**	**Generated output**
	• Configure the DMA channel 0 in 'DmaChannelConfig' container. • Configure the notification function as 'Dma_ErrorInterrupt' in 'DmaResourcePartition/DmaMoveEngine ErrorNotifRoutine'.	`extern void Dma_ErrorInterrupt(uint8 Channel, uint32 Event);`

1.3 File: Dma[_<variant>]_PBcfg.h

The generated file contains the extern declaration of the configuration structure for DMA driver. The file is generated in 'inc' folder.

1.3.1 Extern: Dma_Config [_<variant>]

Table 55 Dma_Config [_<variant>]

Name	Dma_Config [_<variant>]	
Type	Dma_ConfigType	
Description	Extern declaration for the root configuration structure of DMA driver which will be used during initialization.	
Verification method	The generated variable is the extern declaration of the structure is present in Dma[_<variant>]_PBcfg.h file. The <variant> indicates the name of the post-build variant. For a variant-aware configuration the extern variable name is appended with the variant name. For variant-unaware configuration <variant> is ignored.	
Example(s)	**Action**	**Generated output**
	DMA is configured and is variant-aware. Variant name is 'Diesel'	`extern const Dma_ConfigType Dma_Config_Diesel;`
	DMA is configured and is variant-unaware	`extern const Dma_ConfigType Dma_Config;`

Revision history

Major changes since the last revision

Date	Version	Description
2020-10-19	4.0	Document Released
2020-10-16	3.1	- Dma driver chapter moved from MC-ISAR_TC3xx_Config_Verification_Manual_CD.pdf to this document. - DMA_DEINIT_API and DMA_SETPATTERN_API macro added. - Dma_kChannel_<x>_TcsConfigRoot structure updated with pattern matching feature.
2019-07-19	3.0	Document is reviewed and released.
2019-07-18	2.1	Updated for the inclusion of the element 'DmaChTrlEnabled' in the Dma_Config structure. Updates to DmaSourceAddress, DmaDestinationAddress and DmaShadowAddress for the provision to use the variable names instead of addresses.
2018-02-27	1.10.0_2.0	Removed additional '_' in generated output of Table 52 is removed.
2018-02-26	1.10.0_1.0	Released version after incorporating the review comments.
2018-02-22	1.10.0_0.1	Initial release.

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2020-10-19
Published by

Infineon Technologies AG

81726 Munich, Germany

© 2020 Infineon Technologies AG.
All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
Doc_Number

IMPORTANT NOTICE
The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics (“Beschaffenheitsgarantie”) .

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer’s compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer’s products and any use of the product of Infineon Technologies in customer’s applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer’s technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS
Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies’ products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.