MCAL User Manual for Dma

32-bit TriCore™ AURIX™ TC3xx microcontroller

About this document

Scope and purpose

This User Manual is intended to enable users to integrate the Microcontroller Abstraction Layer (MCAL) software for the TriCore™ AURIX™ family of 32-bit microcontrollers.

This document describes responsibilities of integrator in-charge of integrating MCAL software with the basic software (BSW) stack. This document also provides detailed information on safety, configuration and functions along with examples of usage of significant features.

***Note**: Detailed information about package installation, safety and other generic information that are common across all modules are provided in MCAL User Manual General.*

Intended audience

This document is intended for anyone using the Dma module of the TC3xx MCAL software.

Document conventions

Table 1 Conventions

Convention	Explanation
Bold	Emphasizes heading levels, column headings, table and figure captions, screen names, windows, dialog boxes, menus, sub-menus
Italics	Denotes variable(s) and reference(s)
`Courier New`	Denotes APIs, functions, interrupt handlers, events, data types, error handlers, file/folder names, directories, command line inputs, code snippets
>	Indicates that a cascading sub-menu opens when you select a menu item
[cover parentID=<alpha numeric value>]	Used for traceability completeness. Reader should ignore these.

Reference documents

This User Manual should be read in conjunction with the following documents:

- AURIX™ TC3xx MCAL User Manual General

Table of contents

About this document . . . 1
Table of contents . . . 2
1 Dma driver . . . 6
1.1 User information . . . 6
1.1.1 Description . . . 6
1.1.2 Hardware-software mapping . . . 6
1.1.2.1 DMA: primary hardware peripheral . . . 7
1.1.2.2 SRC: dependent hardware peripheral . . . 8
1.1.2.3 SCU: dependent hardware peripheral . . . 9
1.1.3 File structure . . . 9
1.1.3.1 C file structure . . . 9
1.1.3.2 Code generator plugin files . . . 11
1.1.4 Integration hints . . . 12
1.1.4.1 Integration with AUTOSAR stack . . . 12
1.1.4.2 Multicore and Resource Manager . . . 15
1.1.4.3 MCU support . . . 15
1.1.4.4 Port support . . . 16
1.1.4.5 DMA support . . . 16
1.1.4.6 Interrupt connections . . . 16
1.1.4.7 Example usage . . . 18
1.1.5 Key architectural considerations . . . 23
1.1.5.1 Usage of General Purpose Software Request (GPSR) . . . 23
1.2 Assumptions of Use (AoU) . . . 24
1.3 Reference information . . . 27
1.3.1 Configuration interfaces . . . 27
1.3.1.1 Container: CommonPublishedInformation . . . 27
1.3.1.1.1 ArMajorVersion . . . 27
1.3.1.1.2 ArMinorVersion . . . 28
1.3.1.1.3 ArPatchVersion . . . 28
1.3.1.1.4 ModuleId . . . 29
1.3.1.1.5 Release . . . 29
1.3.1.1.6 SwMajorVersion . . . 30
1.3.1.1.7 SwMinorVersion . . . 30
1.3.1.1.8 SwPatchVersion . . . 31
1.3.1.1.9 VendorId . . . 31
1.3.1.2 Container: Dma . . . 31
1.3.1.3 Container: DmaChannelConfig . . . 32
1.3.1.3.1 DmaChannelAssignedPartition . . . 32
1.3.1.3.2 DmaChannelId . . . 32

Table of contents

1.3.1.3.3 DmaChannelNotification . . . 33
1.3.1.3.4 DmaChannelNumTransactionSet . . . 33
1.3.1.3.5 DmaChannelUser . . . 34
1.3.1.3.6 DmaErrorNotification . . . 35
1.3.1.3.7 DmaTcsInterruptTransactionLoss . . . 35
1.3.1.4 Container: DmaChannelTransactionSet . . . 36
1.3.1.4.1 DmaCLLNextIndex . . . 36
1.3.1.4.2 DmaNextTcsIndex . . . 36
1.3.1.4.3 DmaPatternMode . . . 37
1.3.1.4.4 DmaTcsAppendTimeStamp . . . 38
1.3.1.4.5 DmaTcsAutoStartEnable . . . 38
1.3.1.4.6 DmaTcsCircularBufferDestinationEnable . . . 39
1.3.1.4.7 DmaTcsCircularBufferDestinationLength . . . 39
1.3.1.4.8 DmaTcsCircularBufferSourceEnable . . . 40
1.3.1.4.9 DmaTcsCircularBufferSourceLength . . . 41
1.3.1.4.10 DmaTcsDaisyChaining . . . 42
1.3.1.4.11 DmaTcsDataTransferInterrupt . . . 43
1.3.1.4.12 DmaTcsDestinationAddress . . . 43
1.3.1.4.13 DmaTcsDestinationAddressModificationFactor . . . 44
1.3.1.4.14 DmaTcsDestinationAddressMovement . . . 45
1.3.1.4.15 DmaTcsDoubleBuffer . . . 46
1.3.1.4.16 DmaTcsHardwareTrigger . . . 46
1.3.1.4.17 DmaTcsIndex . . . 47
1.3.1.4.18 DmaTcsInterruptDataTransfer . . . 47
1.3.1.4.19 DmaTcsInterruptDataTransferThreshold . . . 48
1.3.1.4.20 DmaTcsInterruptDestAddressWrap . . . 49
1.3.1.4.21 DmaTcsInterruptSourceAddressWrap . . . 49
1.3.1.4.22 DmaTcsMoveLength . . . 50
1.3.1.4.23 DmaTcsReferenceAddressCrc . . . 50
1.3.1.4.24 DmaTcsReferenceDataCrc . . . 51
1.3.1.4.25 DmaTcsShadowRegisterConfiguration . . . 51
1.3.1.4.26 DmaTcsSourceAddress . . . 53
1.3.1.4.27 DmaTcsSourceAddressModificationFactor . . . 53
1.3.1.4.28 DmaTcsSourceAddressMovement . . . 54
1.3.1.4.29 DmaTcsSwapDataCRCByteOrder . . . 55
1.3.1.4.30 DmaTcsTransactionLength . . . 55
1.3.1.4.31 DmaTcsTransferLength . . . 56
1.3.1.4.32 DmaTcsTriggerFrequency . . . 57
1.3.1.4.33 DmaUserHeaderFileWithExternDeclarations . . . 57
1.3.1.5 Container: DmaGeneral . . . 58
1.3.1.5.1 DmaBufferSwitchApi . . . 58
1.3.1.5.2 DmaChDeInitApi . . . 58

Table of contents

1.3.1.5.3 DmaDataPendingApi 59
1.3.1.5.4 DmaDeinitApi 59
1.3.1.5.5 DmaDevErrorDetect 60
1.3.1.5.6 DmaGetVersionInfoApi 60
1.3.1.5.7 DmaInitCheckApi 61
1.3.1.5.8 DmaInitDeInitApiMode 62
1.3.1.5.9 DmaMaxTransactionSetPerChannel 62
1.3.1.5.10 DmaMultiCoreErrorDetect 63
1.3.1.5.11 DmaRuntimeApiMode 63
1.3.1.5.12 DmaSafetyEnable 64
1.3.1.5.13 DmaSuspendApi 64
1.3.1.5.14 DmaTriggerApi 65
1.3.1.6 Container: DmaMoveEngineConfig 65
1.3.1.6.1 DmaMEDestinationErrorInterrupt 66
1.3.1.6.2 DmaMELinkedListErrorInterrupt 66
1.3.1.6.3 DmaMESourceErrorInterrupt 67
1.3.1.7 Container: DmaPatternConfig 67
1.3.1.7.1 DmaChannelForPattern0 67
1.3.1.7.2 DmaChannelForPattern1 68
1.3.1.7.3 DmaPattern0 69
1.3.1.7.4 DmaPattern1 70
1.3.1.8 Container: DmaResourcePartition 70
1.3.1.8.1 DmaPermittedBusMaster 70
1.3.1.8.2 DmaResourcePartitionBusMode 71
1.3.2 Functions - Type definitions 72
1.3.2.1 Dma_ChannelNotificationPtrType 72
1.3.2.2 Dma_ConfigType 72
1.3.2.3 Dma_ConfigUpdateType 72
1.3.2.4 Dma_CrcType 73
1.3.2.5 Dma_ErrorNotificationPtrType 74
1.3.2.6 Dma_EventsType 74
1.3.2.7 Dma_MoveEngineListType 75
1.3.3 Functions - APIs 75
1.3.3.1 Dma_DeInit 75
1.3.3.2 Dma_Init 76
1.3.3.3 Dma_IsInitDone 77
1.3.3.4 Dma_ChInit 78
1.3.3.5 Dma_ChUpdate 79
1.3.3.6 Dma_ChDeInit 80
1.3.3.7 Dma_ChTransferFreeze 81
1.3.3.8 Dma_ChTransferResume 82
1.3.3.9 Dma_ChEnableHardwareTrigger 83

Table of contents

1.3.3.10 Dma_ChDisableHardwareTrigger 84
1.3.3.11 Dma_ChStartTransfer 85
1.3.3.12 Dma_ChStopTransfer 86
1.3.3.13 Dma_ChGetRemainingData 86
1.3.3.14 Dma_ChSwitchBuffer 87
1.3.3.15 Dma_GetEvents 88
1.3.3.16 Dma_ChStatusClear 89
1.3.3.17 Dma_ChInterruptEnable 90
1.3.3.18 Dma_ChInterruptDisable 91
1.3.3.19 Dma_GetVersionInfo 92
1.3.3.20 Dma_MEStatusClear 93
1.3.3.21 Dma_InitCheck 94
1.3.3.22 Dma_GetCrcValue 95
1.3.3.23 Dma_GetCurrentTimeStamp 96
1.3.3.24 Dma_IsChannelInitDone 97
1.3.3.25 Dma_SetPattern 98
1.3.4 Notifications and Callbacks 99
1.3.5 Scheduled functions 99
1.3.6 Interrupt service routines 99
1.3.6.1 Dma_ChInterruptHandler 99
1.3.6.2 Dma_MEInterruptDispatcher 100
1.3.7 Callout 101
1.3.8 Errors Handling 101
1.3.9 Deviations and limitations 103
1.3.9.1 Deviations 103
1.3.9.1.1 Software specification deviations 103
1.3.9.1.2 AMDC Violations 104
1.3.9.1.3 VSMD Violations 104
1.3.9.2 Limitations 104
Revision history 106
Disclaimer 107

1 Dma driver

1.1 User information

1.1.1 Description

The DMA driver is responsible for providing the services and configuration options, for performing the DMA operations using the AURIX DMA hardware. Services for initialization, starting, stopping and updating the DMA channels are provided by the driver. The driver is designed as a post-build variant and it is possible to generate a hex file specifically for a desired configuration.

1.1.2 Hardware-software mapping

This section describes the system view of the DMA driver and peripherals administered by it.

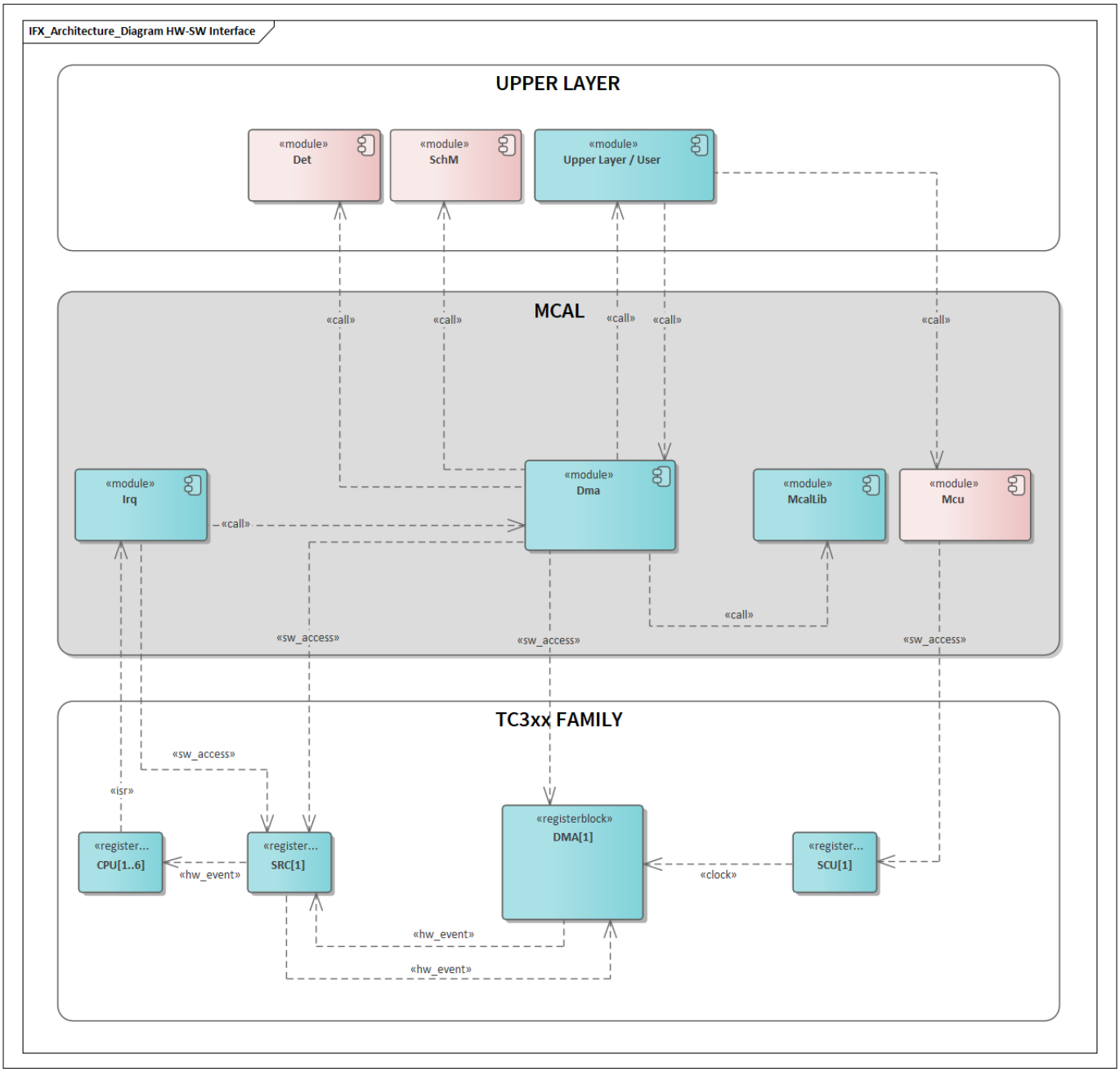

Figure 1 Mapping of hardware-software interfaces

1.1.2.1 DMA: primary hardware peripheral

Hardware functional features

The DMA driver uses the DMA channels and the DMA move engine of the DMA IP for DMA transactions. The DMA channel holds the transfer characteristics such as the source address, destination address, transfer length, triggering criteria and so on. The move engine is a bus master and performs the physical movement of data from the source memory (SRAM, peripherals) to the destination memory (SRAM, peripherals) based on the transfer characteristics. The transfer characteristics can be configured using the configuration tools. In addition, the transaction control set can be updated through the driver APIs.

The key hardware functional features used by the driver are:

- Double buffering operations - source and destination side

- Circular buffer - source and destination
- Linked list, accumulated linked list, safe linked list and conditional linked list
- Channel trigger - by the software and the hardware
- Daisy chaining
- CRC calculations for address and data
- Time stamping of the transactions
- Interrupts from channel and resource partitions
- Pattern matching

The unsupported features of the DMA hardware are:

- Activation of error interrupts using Error Interrupt Set (ERRINTRr) register

Users of the hardware

The DMA driver exclusively utilizes the DMA IP. The peripherals which require data transfer from one memory location to another should use the DMA driver APIs to perform the transfers.

Hardware diagnostic features

The SMU alarms configured for the DMA IP are not monitored by the DMA driver.

Hardware events

The DMA driver uses the following hardware events from the DMA IP:

- DMA channel interrupt service requests
- DMA resource partition interrupt service requests

In addition, the DMA being a service provider, can take up the service requests from other interrupt sources as well.

1.1.2.2 SRC: dependent hardware peripheral

Hardware functional features

The DMA driver depends on the interrupt router for raising an interrupt to the CPU based on the channel interrupt events and resource partition error interrupt events. DMA also depends on the interrupt router for routing the events to DMA, which are flagged by peripherals as data transfer requests.

Users of the hardware

The interrupt router is configured either by the IRQ driver or the user software. However, the SRC registers for the GPSR (General Purpose Service Request) are accessed by the DMA driver, to route the error interrupt requests to the appropriate core in a multicore environment.

Hardware diagnostic features

The SMU alarms configured for interrupt router are not monitored by the DMA driver.

Hardware events

The interrupt events raised by the interrupt router are serviced by the CPU or the DMA. The DMA driver provides interrupt handlers as software interfaces, which shall be invoked from the ISR.

1.1.2.3 SCU: dependent hardware peripheral

Hardware functional features

The DMA driver depends on the SCU IP for the clock and ENDINIT functionalities. For functioning, the driver requires the fSPB clock signal.

Users of the hardware

The SCU IP supplies clock for all the peripherals and the MCU driver is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, update to all the ENDINIT protected registers is performed using the MCALLIB APIs.

Hardware diagnostic features

The SMU alarms configured for the SCU IP are not monitored by the DMA driver.

Hardware events

Hardware events from the SCU are not used by the DMA driver.

1.1.3 File structure

1.1.3.1 C file structure

This section provides details of the C files of the DMA driver.

1 Dma driver

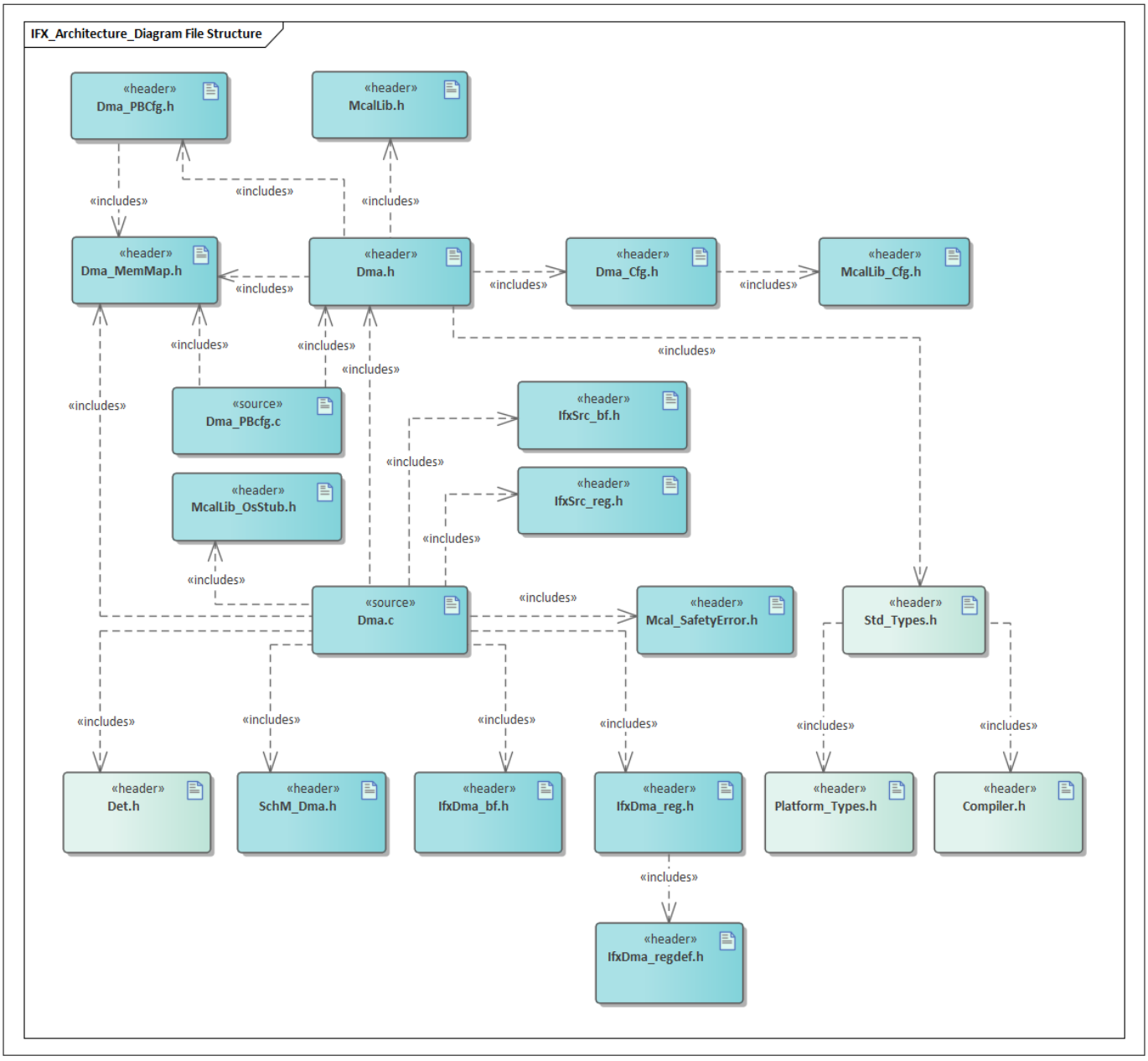

Figure 2 Dma_File_Structure-1.png

Table 2 C file structure

File name	Description
`Compiler.h`	Provides abstraction from compiler-specific keywords
`Det.h`	Provides the exported interfaces of Development Error Tracer
`Dma.c`	File (static) containing implementation of APIs
`Dma.h`	Header file (static) defining prototypes of data structures and APIs
`Dma_Cfg.h`	Header file (generated) containing constants and pre-processor macros
`Dma_MemMap.h`	File containing the memory section definitions used by the DMA driver
`Dma_PBCfg.h`	Post-Build header file for DMA driver

(table continues...)

Table 2 (continued) C file structure

File name	Description
Dma_PBcfg.c	File (generated) containing objects to data structures
IfxDma_bf.h	SFR header file for DMA
IfxDma_reg.h	SFR header file for DMA
IfxDma_regdef.h	SFR header file for DMA
IfxSrc_bf.h	SFR header file for Interrupt Controller
IfxSrc_reg.h	SFR header file for Interrupt Controller
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB.
McalLib_Cfg.h	Generated header file providing information on number of cores, DSPR, PSPR (start and end addresses) and system and backup clock information.
McalLib_OsStub.h	McalLib_OsStub.h provides macros to support user mode of Tricore. This shall be included by other drivers to call OS APIs.
Mcal_SafetyError.h	Header file containing the prototype of the API for reporting safety-related errors
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
SchM_Dma.h	Schedule manager header file for critical section management
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

1.1.3.2 Code generator plugin files

This section provides details of the code generator plugin files of the DMA driver.

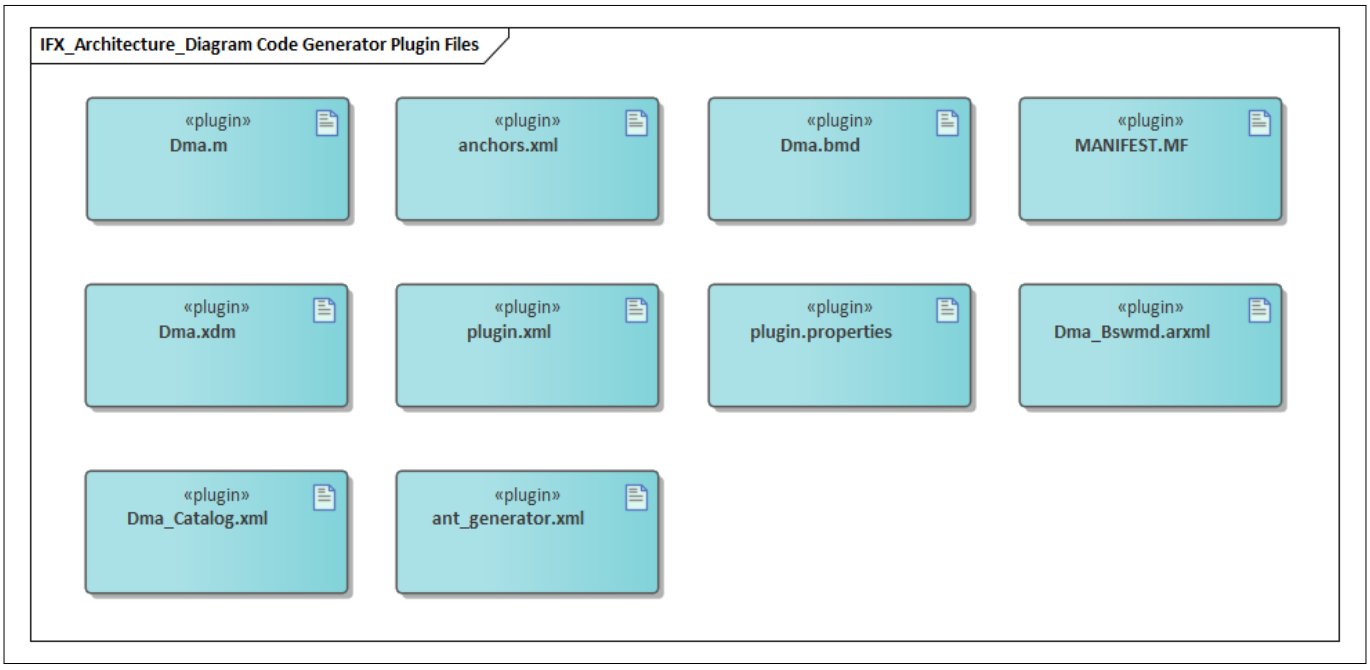

Figure 3 Dma_Code_Generator_Plugin_Files-1.png

Table 3 Code generator plugin files

File name	Description
Dma.bmd	AUTOSAR format XML data model schema file (for each device)
Dma.m	Code template macro file for DMA driver
Dma.xdm	Tresos format XML data model schema file
Dma_Bswmd.arxml	AUTOSAR format module description file
Dma_Catalog.xml	AUTOSAR format catalog file
MANIFEST.MF	Tresos plugin support file containing the meta-data for DMA driver
anchors.xml	Tresos anchors support file for the DMA driver
ant_generator.xml	Tresos support file to generate and rename multiple post-build configurations when using variation point
plugin.properties	Tresos plugin support file for the DMA driver
plugin.xml	Tresos plugin support file for the DMA driver

1.1.4 Integration hints

This section lists the key points, that an integrator or user of the DMA driver must consider.

1.1.4.1 Integration with AUTOSAR stack

This section lists the modules that are not part of MCAL, but are required to integrate the DMA driver

- **EcuM**

  The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of ECU. Specifically, in the context of MCAL the EcuM is used for initialization and de-initialization of the software drivers. The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase.

- **Memory mapping**

  Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the re-locatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the Dma_MemMap.h file.

  The Dma_MemMap.h file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements

are relocated to the correct memory region. A sample implementation listing the memory-section macros is shown as follows:

```
/**** GLOBAL RAM DATA -- NON-CACHED LMU ****/
#if defined DMA_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
 /*****User pragmas here for Non-cached LMU*****/
 #undef DMA_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
 #undef MEMMAP_ERROR
#elif defined DMA_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
 #ifdef _TASKING_C_TRICORE_
 /*****User pragmas here for Non-cached LMU*****/
 #undef DMA_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
 #undef MEMMAP_ERROR

/**** CORE[x] CONFIG DATA -- PF[x] ****/ /*[x]=0..5*/
#elif defined DMA_START_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
 /*****User pragmas here for PF[x]*****/
 #undef DMA_START_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
 #undef MEMMAP_ERROR
#elif defined DMA_STOP_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
 /*****User pragmas here for PF[x]*****/
 #undef DMA_STOP_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
 #undef MEMMAP_ERROR

/**** CORE[x] CONFIG DATA WITH 256bit ALIGNMENT -- PF[x] ****/ /*[x]=0..5*/
#elif defined DMA_START_SEC_CONFIG_DATA_ASIL_B_CORE0_256
 /*****User pragmas here for PF[x], with 256 bit alignment*****/
 #undef DMA_START_SEC_CONFIG_DATA_ASIL_B_CORE0_256
 #undef MEMMAP_ERROR
 #elif defined DMA_STOP_SEC_CONFIG_DATA_ASIL_B_CORE0_256
 /*****User pragmas here for PF[x], with 256 bit alignment*****/
 #undef DMA_STOP_SEC_CONFIG_DATA_ASIL_B_CORE0_256
 #undef MEMMAP_ERROR

/**** CODE -- PF[x] ****/
#elif defined DMA_START_SEC_CODE_ASIL_B_GLOBAL
 /*****User pragmas here for PF[x]*****/
 #undef DMA_START_SEC_CODE_ASIL_B_GLOBAL
 #undef MEMMAP_ERROR
#elif defined DMA_STOP_SEC_CODE_ASIL_B_GLOBAL
 /*****User pragmas here for PF[x]*****/
 #undef DMA_STOP_SEC_CODE_ASIL_B_GLOBAL
 #undef MEMMAP_ERROR

#endif

#if defined MEMMAP_ERROR
#error "Dma_MemMap.h, wrong pragma command"
#endif
```

- **DET**

The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the BSW modules. The DMA driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the DMA driver must process all the errors reported to the DET module through the `Det_ReportError()` API.

The `Det.h` and `Det.c` files are provided in the MCAL package as a stub code and are required to be replaced with a complete DET module during the integration phase.

- **DEM**

  The DEM module is not required for the integration of the DMA driver.

- **SchM**

  The SchM module is a part of the RTE that manages the BSW scheduler. The DMA driver uses the exclusive areas defined in `SchM_Dma.h` file to protect the SFRs and variables from concurrent accesses from different threads. The SchM identified for the DMA driver is:

  - ChannelConfigUpdate

  The `SchM_Dma.h` and `SchM_Dma.c` files are provided in the MCAL package as an example code and are required to be updated by the integrator. The user must implement the SchM functions defined by the DMA driver as **suspend/resume** of interrupts for the CPU on which the API is invoked. A sample implementation of the SchM functions is shown as follows:

```
/**** Sample implementation of SchM_Dma.c ****/
#include "Os.h"

void SchM_Enter_Dma_ChannelConfigUpdate(void)
{
 /* Start of Critical Section */
 SuspendAllInterrupts(); /* Suspend CPU core interrupt */
}

void SchM_Exit_Dma_ChannelConfigUpdate(void)
{
 /* End of Critical Section */
 ResumeAllInterrupts(); /* Resume CPU core interrupt */
}
```

- **Safety error**

  The DMA driver reports all the detected safety errors through the `Mcal_ReportSafetyError()` API.

  The driver performs only detection and reporting of the safety errors. The handling of the reported errors shall be done by the user. The `Mcal_ReportSafetyError()` API is provided in the `Mcal_SafetyError.c` and `Mcal_SafetyError.h` files as a stub code, and must be updated by the integrator to handle the reported errors.

  Note: All DET errors are also reported as safety errors (error code used is same as DET).

- **Notifications and callbacks:**

  The DMA driver does not implement any notifications. However, it does report the interrupts raised by the DMA channels and error interrupts from the DMA Resource Partitions through notification functions. These notification functions can be configured by the user in the EB Tresos tool for each DMA channel and DMA Resource Partitions separately.

  Note: The DMA interrupts shall be assigned the interrupt priority higher than the interrupts from the users of DMA.

- **Operating system (OS):**

The OS or application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of interrupts must also be managed by the OS or the application.

The OS files provided by the MCAL package are only for example purpose and must be updated by the integrator with the actual OS files for the desired function.

Note: The DMA driver updates the SRC registers of GPSR (`MODULE_SRC.GPSR.GPSR0.SR0`) to clear the pending interrupt requests.:

1.1.4.2 Multicore and Resource Manager

The DMA driver supports execution of its APIs in simultaneously from all the CPU cores. The user has to allocate resources of the DMA to the CPU cores at pre-compile time using the Resource Manager module. The following are the key points to be considered with respect to the multicore functionality in the driver:

- Initialization of the DMA driver should be invoked from each CPU core, which intends to use the services of the DMA driver. Initialization from logical master core must be completed prior to invoking the initialization services from other slave cores. The slave cores can execute initialization in parallel. When all the DMA resources are allocated to the master core, invoking the initialization from the master core alone is sufficient.
- DMA channels can be allocated to the any of the CPU cores. For example, DMAchannel-0 and DMAchannel-55 can be allocated to core-1; DMAchannel-21 and DMAchannel-35 can be allocated to core-2; DMAchannel-10 and DMAchannel-101 can be allocated to core-0 and so on.
- Any configured DMA channel, if unallocated in the Resource Manager, will be directly allocated to the master core.
- It must be ensured that the DMA channel number passed as a parameter while invoking a DMA API, belong to the same core on which the API was invoked.
- DETs and/or safety errors are raised in case APIs are invoked when there is a mismatch of core and DMA channel assignments.
- Channel interrupts raised by a DMA channel must be serviced by the CPU core to which the DMA channel has been allocated to. However, the Resource Partition interrupts must be serviced by the master CPU core.
- Locating constants, variables and configuration data to the correct memory space should be done by the user. Memory sections are marked GLOBAL (common to all cores) and CORE[x] (specific to a CPU core). The following should be considered by the user to ensure better performance of the driver:

  Code section :

  The executable code of the DMA driver is placed under single MemMap section. It can be relocated to any PFlash region.

  Data section :

  The RAM variable memory sections marked as specific to a particular core should be re-located to the DSPR/DLMU of the same core. Those marked as global should be relocated to the non-cached LMU region.

  Configuration data and constants :

  The configuration data section marked as specific to a particular core should be re-located to the PFlash of the same core. Those marked as global should be relocated to the PFlash of the master core.

  Note: Relocating code, data or constants to a distant memory space would impact the execution timings.

  Note: If the driver operates from a single (master) core, all the sections may be relocated to the PFlash or DSPR of the same CPU core.

1.1.4.3 MCU support

The DMA driver is dependent on the MCU driver for the clock configuration. The initialization of the DMA driver must be started only after completing the MCU initialization.

1.1.4.4 Port support

The DMA driver does not use any services provided by the PORT driver.

1.1.4.5 DMA support

Following measures are for better handling of DMA driver.

ESM - DMA Error handling and Supervision

For error reporting, DMA provides configurable error callback DmaErrorNotification to respective user or driver.

For error handling and supervision, following steps are recommended:

1. Reset the channel using Dma_ChStopTransfer API

2. Initialize the channel using Dma_ChInit API

Note: The channel would have reinitialized to the initialization values which were provided during configuration.

Note: The provided measure is a generic recommended measure. The user can decide on using other measures, which are deemed sufficient as per the usecase.

Efficient DMA interrupt handling

For efficient DMA interrupt handling and to avoid General Purpose Service Requests (GPSR) routing, following steps are recommended:

1. Allocate a core to a one or more Resource Partition(RP).

2. Configure the Type of Service (TOS) and the RP interrupt to the allocated core.

3. Allocate channels of the particular core to the allocated RPs

Note: Assuming 1 core to single RP, then the above measure is only possible for four cores (only four RPs in the hardware). If more cores need support, GPSR routing would be required.

1.1.4.6 Interrupt connections

The interrupt connections of the DMA driver are described in this section:

DMA channel ISR

Each DMA channel has a dedicated interrupt. These interrupts are raised signifying a channel transfer completion, source wrap or a destination wrap. The channel interrupts may be enabled in the driver configuration. The integrator must ensure that the Service Request Control(SRC) registers of these interrupts are configured appropriately.

DMA RP error ISR

Each Resource Partition has a dedicated interrupt line. These are the error interrupts possible during the DMA operations. These interrupts may be enabled in the driver configuration. In a multicore environment, the error interrupts are routed to the appropriate cores using a GPSR0 interrupt. The integrator must ensure that the SRC registers of all these interrupts are configured appropriately.

Note: If DMA is to be used in polling mode, the DMA interrupts needs to be disabled in DMA SRC itself, otherwise the status flags will be cleared in the interrupt handlers.

DMA spurious interrupt handling

DMA driver does not report spurious interrupts.

For customers to identify spurious interrupts to DMA driver, below are the recommendations:

1) Configure DMA Channel TRL events if expected. Exception is DMA channels allocated to SPI driver which handle the TRL events internally.

2) Configure callback to DMA channels. (Note: For MCAL drivers like SPI, CRC, it is to be configured as recommended by the respective drivers.)

If a spurious interrupt is routed to DMA driver, it would exit the ISR without any callbacks since below conditions are not met :

1) No interrupt reasons set in the hardware (DMA_ME<x>_ERRSR where x=0,1 or CHCSR channel SFR) and hence, callback routines are not invoked.

2) Configured TRL is not enabled for the particular channel and hence, callback routine is not invoked.

The user can monitor the interrupt trigger from the hardware and the absence of an expected driver callback, which can be a mechanism to detect the spurious interrupts.

Note: To ensure timely service of DMA interrupts, user should ensure assignment of appropriate priorities to the DMA interrupts as needed in the system, to ensure that the interrupt servicing rate is greater than the interrupt generation rate. If this condition is not met, interrupt notifications to the user can be missed out or interrupt can report 'DMA_EVENT_CH_UNKNOWN_EVENT' to the user.

1.1.4.7 Example usage

The following are some of the key use cases of the DMA driver:

Note: For additional information, refer to the comments in the code snippets.

Initialization of DMA driver

The driver can be initialized by invoking the `Dma_Init()` API. During the invocation, the configuration structure pointer is passed as an input parameter.

Note: The driver gets initialized for the core (along with the channels allocated to the core) in which the API was invoked. The initialization should be carried out for each core where the driver is going to be used.

```
/* Include the header files */
#include "Dma.h"
#include "Mcu.h"

/* MCU Initialization */
Mcu_Init(&Mcu_Config);
Mcu_InitClock(0U);
while(Mcu_GetPllStatus() != MCU_PLL_LOCKED);
Mcu_DistributePllClock();

/* Initialize the DMA driver*/
Dma_Init(&Dma_Config);
/* Driver is initialized for the current core */
```

De-initializing DMA channels

A DMA channel can be selectively de-initialized if needed. The `Dma_ChDeInit()` API is used for the deinitialization.

```
/*
 * Configuration values mandatory for below code snippet:-
 * DmaDeinitApiConfiguration = TRUE
 * -> This is needed to invoke the channel de-init API.
 */

/* De-initialize the DMA channel 12 */
Dma_ChDeInit(12);
```

Starting a DMA channel

After the initialization is complete, the desired DMA channel can be started using the `Dma_ChStartTransfer()` API. Provide the channel number as the parameter for this API.

```
/* Start the DMA channel 5 */
Dma_ChStartTransfer(5);
```

Enabling the hardware trigger for the DMA channel

To enable the DMA channels to receive the interrupts from different sources, the `Dma_ChEnableHardwareTrigger()` API is used.

```
/*
 * Configuration values mandatory for below code snippet:-
 * DmaTriggerApiConfiguration = TRUE
 * -> This is needed to invoke the hardware trigger APIs.
 */

/* Enable the hardware triggers to the DMA channel 10 */
Dma_ChEnableHardwareTrigger(10);
```

Updating the channel settings during runtime

The driver provides the `Dma_ChUpdate()` API to update the following configuration items at the runtime – Source Address, Destination Address, Shadow Address, Address CRC, Data CRC and DMA channel configuration registers.

```
/* Perform an update of the source and destination addresses
 * of channel 10 using the Dma_ChUpdate API. */

/* The source and destination locations */
uint32 ChSourceAddress;
uint32 ChDestinationAddress;

/* Here, 'DmaChannelConfigVar' is a structure of type Dma_ConfigUpdateType,
 * holding the channel configuration to be used for updating the channel */
Dma_ConfigUpdateType DmaChannelConfigVar;

/* Step 1: Set the source and destination address pointers.
 * Step 2: Set the corresponding bit fields indicating that an update is needed
 * for the source and destination addresses */
DmaChannelConfigVar.SourceAddress = &ChSourceAddress;
DmaChannelConfigVar.DestAddress = &ChDestinationAddress;
DmaChannelConfigVar.UpdateAddressCrc = 0;
DmaChannelConfigVar.UpdateConfig = 0;
DmaChannelConfigVar.UpdateControlAdicr = 0;
DmaChannelConfigVar.UpdateControlChcsr = 0;
DmaChannelConfigVar.UpdateDataCrc = 0;
DmaChannelConfigVar.UpdateDestAddress = 1;
DmaChannelConfigVar.UpdateShadowConfig = 0;
DmaChannelConfigVar.UpdateSourceAddress = 1;

/* Perform the update using the driver API */
Dma_ChUpdate(10, &DmaChannelConfigVar, NULL_PTR);
```

Setting up of the linked list during configuration

The linked list for the DMA transactions can be setup during the configuration in Tresos. An example sequence is provided as follows, depicting the relevant parameters:

- **Step 1:** Configure the number of Transaction Control Sets (TCS) needed for the DMA channel.

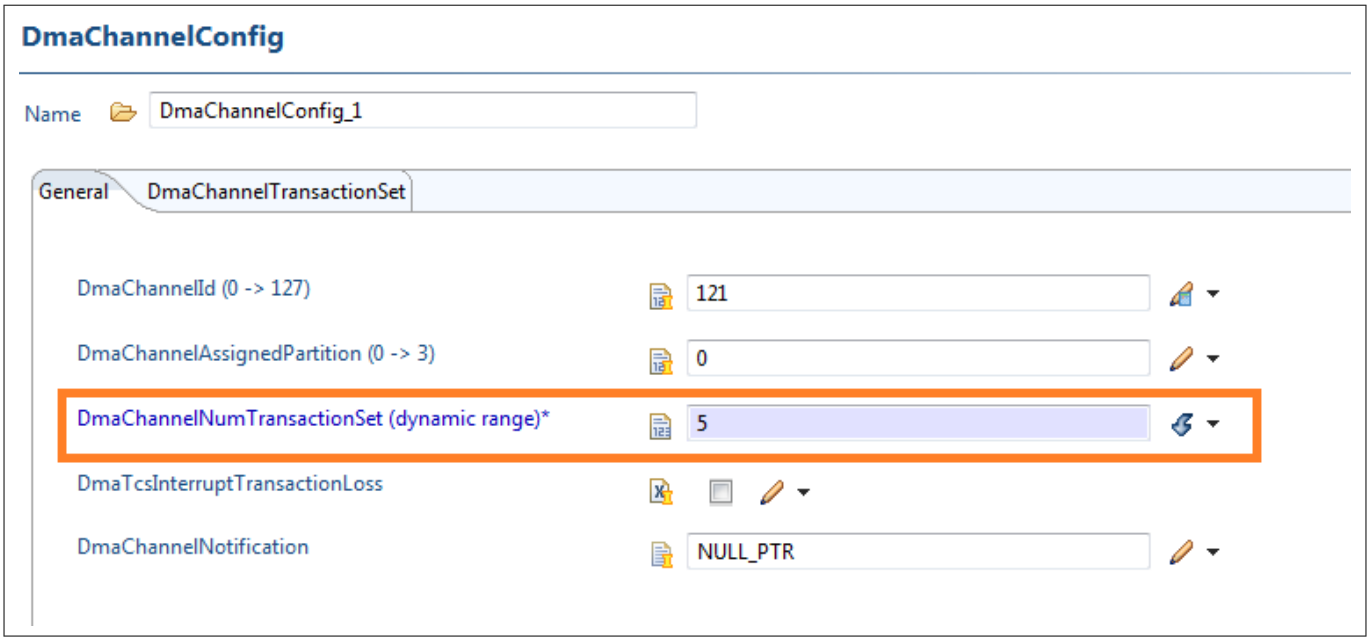

Figure 4 **Step 1: Configure the number of Transaction Control Sets (TCS) needed for the DMA channel.**

- **Step 2:** Configure and add the TCSes to the configuration.

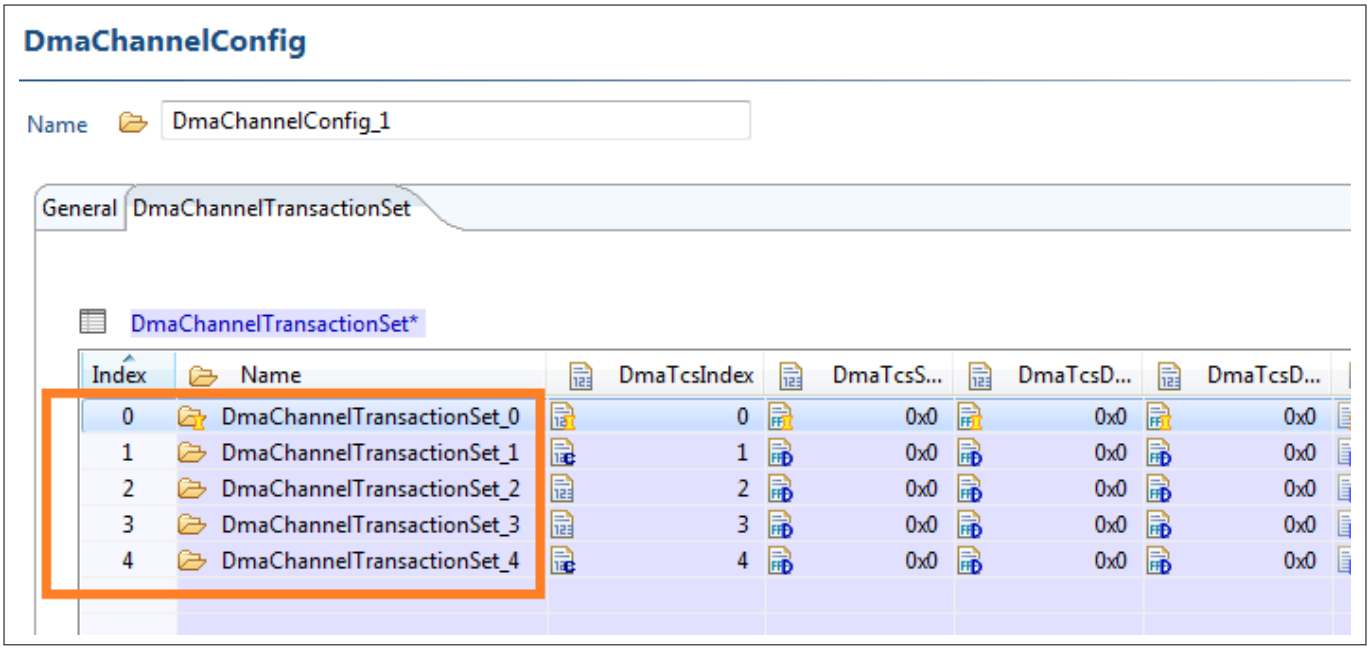

Figure 5 **Step 2: Configure and add the TCSes to the configuration.**

- **Step 3:** Configure the order of the TCSes in Tresos.

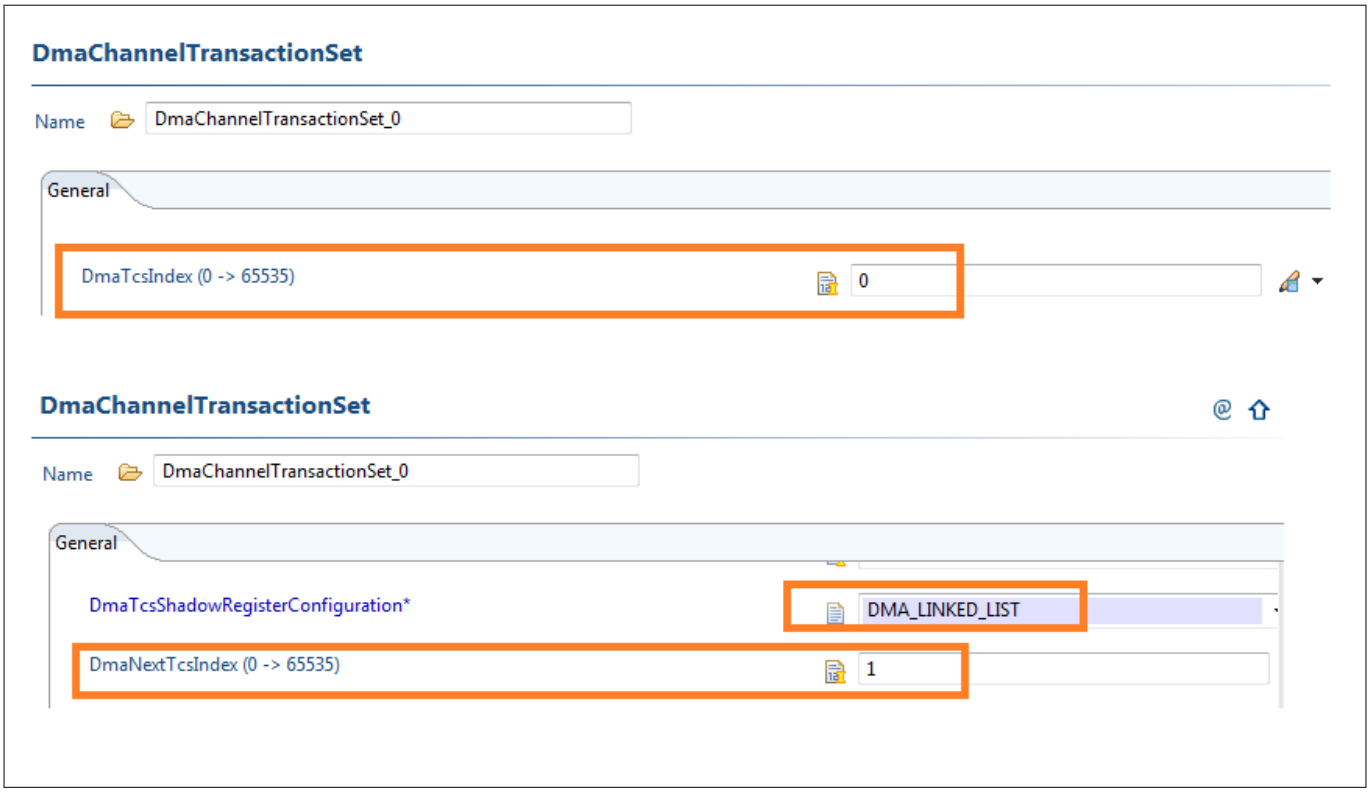

Figure 6 Step 3: Configure the order of the TCSes in Tresos.

Add the channel-specific configurations (source address, destination address and so on) to the TCSes and the channel can be used for the linked list.

Setting up of the linked list during runtime

The linked list for the DMA transactions can be setup during the runtime also. An example sequence is provided as follows, depicting the relevant parameters:

```
/*
 * Configuration values mandatory for below code snippet:-
 * DmaTcsShadowRegisterConfiguration = DMA_LINKED_LIST or
 * DMA_ACCUMULATED_LINKED_LIST or
 * DMA_SAFE_LINKED_LIST or
 * DMA_CONDITIONAL_LINKED_LIST
 * -> This is needed to enable the linked list option.
 */

/* The initial configuration of the channel has to be
 * created using the datatype Dma_ConfigUpdateType.
 * The order of the data to be provided and hints for the data
 * to be entered are mentioned as comments below. */

Dma_ConfigUpdateType Dma_ChannelInitialConfiguration =
{
 /* Provide Source address here. E.g. (uint32)&Dma_Source_01 */
 /* Provide Destination address here. E.g.(uint32)&Dma_Destination_01 */
 /* Provide ADICR register value here */
 /* Provide CHCSR register value here */
 /* Provide CHCFGR register value here */
 /* Provide the contents of the SHADR here.
 This would be the address of the next TCS E.g. (uint32)&Dma_Tcs_01 */
 /* Provide the SDCRC register value here */
 /* Provide the RDCRC register value here */
 /* Provide bit wise information on which of the
 above regsiters are to be updated E.g. 1,1,1,1,1,1,0,0 */
};

/* The further TCSes are to be declared using the
 * DMA provided datatype - Dma_TransactionCtrlSetType
 * The order of the data to be provided and hints for the data
 * to be entered are mentioned as comments below. */

Dma_TransactionCtrlSetType Dma_Tcs_01 =
{
 /* Provide RDCRC register value for TCS */
 /* Provide SDCRC register value for TCS */
 /* Provide SADR register value for TCS */
 /* Provide DADR register value for TCS */
 /* Provide ADICR register value for TCS */
 /* Provide CHCFGR register value for TCS */
 /* Provide SHADR register value for TCS,
 which is the address of the next TCS e.g. (uint32)&Dma_Tcs_02,
 or zero if it is the last TCS. */
 /* Provide CHCSR register value for TCS */
};

Dma_TransactionCtrlSetType Dma_Tcs_02 =
{
```

```
  /* Provide RDCRC register value for TCS */
  /* Provide SDCRC register value for TCS */
  /* Provide SADR register value for TCS */
  /* Provide DADR register value for TCS */
  /* Provide ADICR register value for TCS */
  /* Provide CHCFGR register value for TCS */
  /* Provide SHADR register value for TCS */
  /* Provide CHCSR register value for TCS */
};

/* Update the desired channel with the TCS made ready for the linked list */
Dma_ChUpdate(10, &Dma_ChannelInitialConfiguration, NULL_PTR);
```

1.1.5 Key architectural considerations

1.1.5.1 Usage of General Purpose Software Request (GPSR)

In a multicore environment when DMA error occurs, the error interrupt can get serviced only by any one of the cores. This can create problems, since the cause of the error might be a transaction which was triggered from another core. To circumvent this problem, it is decided to use the General Purpose Software Requests (GPSR). The error interrupts shall be routed to the appropriate core using GPSRs. The error handler, which services the error interrupt finds the core to which the error-prone channel has been allocated, and triggers a GPSR to that core. The GPSR, in turn, calls the notification function, which was configured by the user.

1.2 Assumptions of Use (AoU)

The AoU for the DMA driver are as follows.

- **Spurious Interrupt handling**

DMA driver validates only ME0, ME1 interrupt flags along with channel specific TRL for which the 'DmaTcsInterruptTransactionLoss' configuration parameter is enabled.

User shall have means to determine if a spurious interrupt has occurred and take appropriate actions.

Note: TRL interrupt can be raised for a channel where 'DmaTcsInterruptTransactionLoss' configuration parameter is disabled and user shall determine if this is valid interrupt or not based on their use-cases.

[cover parentID DMA={BFB5045F-6710-485d-9654-0791B82B8E57}]

- **DMA address CRC and data CRC verification**

The user of the DMA driver shall compare the expected CRC with the CRC calculated by the DMA driver to find any mismatch.

[cover parentID DMA={9697DA85-FEC0-4ed0-B41A-98A6EA438E5E}]

- **DMA timestamp feature**

If the timestamp feature of the DMA is used, the user shall compare the timestamp with expected timestamp and take appropriate measures. The user should also allocate a 4 byte space (at the next 32 bit aligned address, after the last byte copied to the destination) for the DMA to store the timestamp.

[cover parentID DMA={1FFDBE80-D8EE-434b-A6D4-BD27986E1DAF}]

- **Double buffering: software switch usage**

When using double buffering in the software switch mode, a TRL event is raised if the transaction completes before the DMA receives a software switch. In a typical use case, the user can set the RROAT bit to 0, so that each transfer would require a trigger.

[cover parentID DMA={946532FE-2B5D-438d-9A1B-01B98321ECA8}]

- **Monitoring the lost transactions of DMA**

The user of the DMA driver is recommended to check for any lost DMA transaction, using the interrupt handlers. If there are any requests lost, the user shall take appropriate actions.

[cover parentID DMA={246B1D83-BF24-4f82-8726-5A01D143233A}]

- **Non-receipt of the interrupts to DMA**

If the DMA channel is configured to receive the interrupts from the hardware as triggers for the transfer/ transaction, the user has to ensure the following:

1. The appropriate interrupt settings should be set to route the interrupts to DMA.

2. The user should also configure and monitor the channel transfer/transaction complete interrupts. These interrupts should be monitored with a timeout mechanism to ensure the correct reception of the hardware interrupts. The user can take appropriate error handling mechanism in the event of a timeout.

[cover parentID DMA={F8377F12-31F4-4e39-B3C8-ACC568522748}]

- **Protection of DMA data and registers**

The user shall ensure that adequate memory protections is made available, so that the DMA global data and registers are not overwritten by any other software running in the system.

[cover parentID DMA={34CC22B0-38D9-436c-B0E3-2438A6DCAE7C}]

- **Usage of APIs for enabling/disabling hardware triggers**

The enable/disable hardware trigger APIs should not be called for a channel which is configured as 'no hardware trigger'

[cover parentID DMA={E7EB27D9-C3BB-4652-910B-4FE393BF8326}]

- **Usage of DMA interrupts**

It is recommended that the user enable the interrupts and configures the respective handlers as well. This would enable the user to detect the events and errors occurring during the DMA transactions and take appropriate measures for handling them.

Note 1: Polling should not be done for detecting the channel events or error events. This is not supported by the driver.

Note 2: The user has to configure and invoke the channel interrupts for the used channels (for detecting channel events), the resource partition interrupts and the GPSR00 interrupt (for detecting the errors).The user has to ensure that the Dma_MEInterruptDispatcher() interrupt handler is called from the GPSR00 interrupt handler.

[cover parentID DMA={D84CE84B-2D57-40bb-8DC0-C9EB17817712}]

- **Usage of Dma_ChUpdate API**

The Dma_ChUpdate API should not be called by the user with invalid TCS contents (reserved bit fields selected, unused features selected, read-only or status bits set). The API also should not be used to update the TCS of an active/pending/halted(frozen) channel. In addition, the user should not modify the pattern matching or conditional linked list settings when using the Dma_ChUpdate API.

[cover parentID DMA={9960DEFF-E8A7-4a6d-A830-45F3BD872C96}]

- **Usage of Dma_IsInitDone API**

The Dma_InitDone API only indicates whether the DMA driver initialization was invoked or not. In case channels were stopped, the user shall not rely or use this API to derive the initialization status of the channel

[cover parentID DMA={FC698CE5-041C-4033-98BB-9D89E5202455}]

- **User/Supervisor mode context usage**

The user shall ensure that the DMA APIs are invoked in the same context for which it is configured to be used. For example, if the DMA is configured to be used in the Supervisor mode, the user should invoke the DMA APIs in the Supervisor mode.

[cover parentID DMA={AD934C16-FC65-4d74-AE9D-47431A53A3E1}]

- **Activation of error interrupts using ERRINTR not supported**

Activation of error interrupts using Error Interrupt Set (ERRINTRr) register is not supported by the driver.

[cover parentID DMA={C11F30B3-B548-4253-9578-993EE367C7BF}]

- **Config Check**

The user shall ensure that the generated configuration structures are correct against the intended GUI configurations.

[cover parentID DMA={C9DFDA58-CB2E-4a50-9AF1-98102D9A3760}]

- **Configuration of GPSR interrupts**

The GPSR00 interrupt is used by the DMA to report the error events of DMA IP. The user should also ensure that the 'Dma_MEInterruptDispatcher' interrupt handler is invoked from the GPSR00 interrupt handler. This is needed for the error events to be notified.

[cover parentID DMA={C24D28F2-FDC4-413e-8E16-416F8B119C5B}]

- **Usage and configuration of interrupts**

When the DMA channels are allocated by the user to any particular core, the user has to ensure that the TOS (in the IRQ configuration) of the corresponding channel interrupts is also assigned to the same core.

[cover parentID DMA={B05BDE05-CEA3-45df-AE66-B87C00256F2B}]

- **Usage of cache and DMA**

If the user requires the DMA to transfer the data to and from the memory, the related memory and the Transaction Control Sets (TCS) should be accessed from the CPU without using the cache. If the cache is used, the consistency of the data cannot be ensured.

[cover parentID DMA={711073AB-D2BA-4145-97EC-F8DEB1DE82A2}]

- **Usage of Init Check feature**

The user has to ensure that the Dma_InitCheck API of the driver is invoked from each core where the DMA initialization is performed. The Dma_InitCheck API invocation should take place after the initialization.
[cover parentID DMA={5407022A-8D01-4646-A8E6-0D4716A6DDD0}]

- **Usage of TRL feature**

If the TRL event has to be reported for a particular channel, the user shall ensure that the ETRL bit is enabled for the channel. This is done by setting the 'DmaTcsInterruptTransactionLoss' configuration parameter in the configuration tool.
[cover parentID DMA={6880790B-2476-4a53-AA66-9CB3565BE1AB}]

- **Software access protection feature usage**

The user has to configure and use the hardware functionality of bus access protection (DMA ACCEN registers) to ensure freedom from interference of the software from different bus masters. The DMA driver provides means to configure these registers through the EB Tresos tool.
[cover parentID DMA={EFBDACD1-8935-40e9-B0D6-1A0870A1A1C5}]

1.3 Reference information

1.3.1 Configuration interfaces

Supported configuration variant: Post-Build

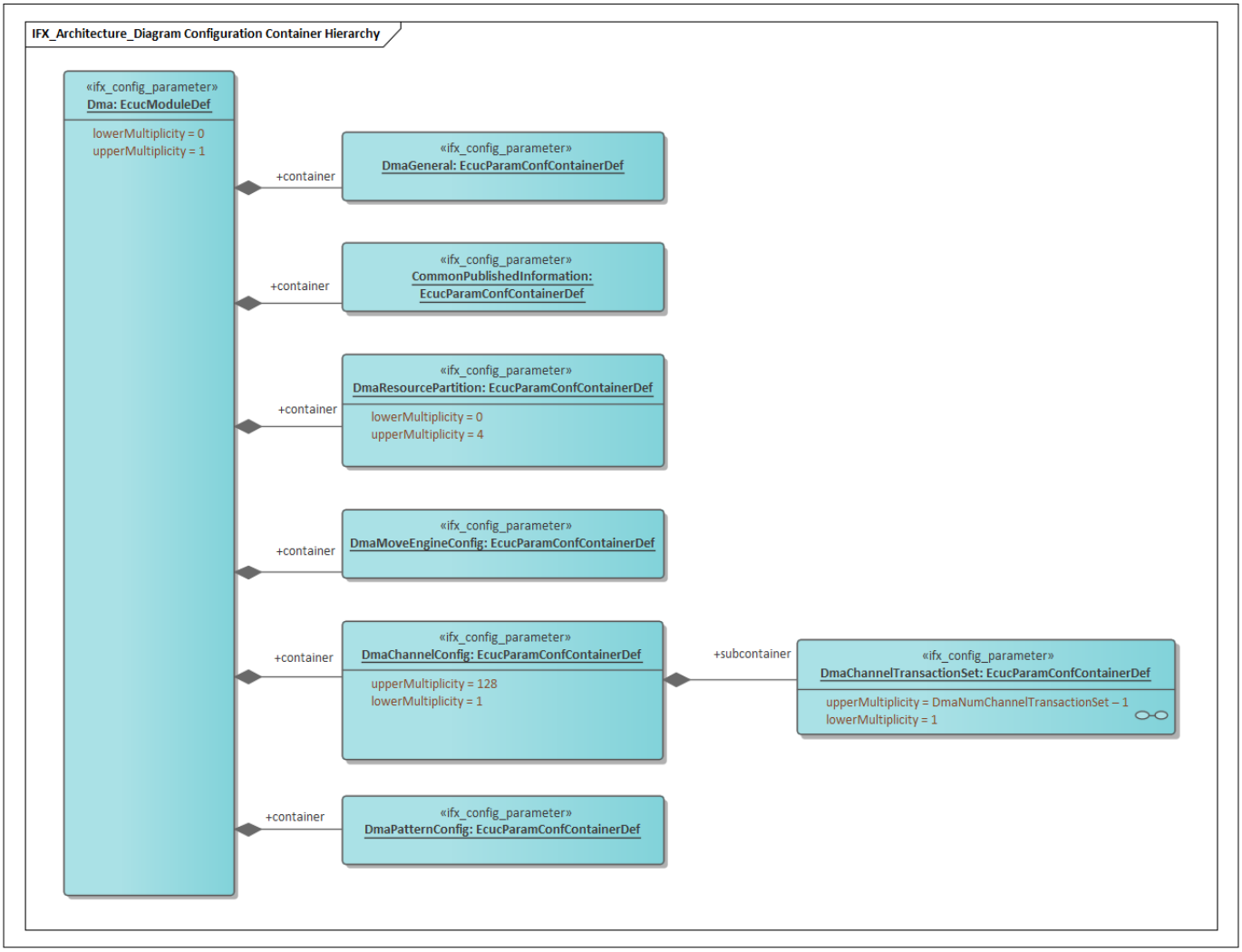

Figure 7 Container hierarchy along with their configuration parameters

1.3.1.1 Container: CommonPublishedInformation

This is the general configuration of DMA driver common container. It contains published information about vendor and versions.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.1.1 ArMajorVersion

Table 4 Specification for ArMajorVersion

Name	`ArMajorVersion`

(table continues...)

Table 4 (continued) Specification for ArMajorVersion

Description	This parameter provides the major version of the AUTOSAR specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the current AUTOSAR version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.2 ArMinorVersion

Table 5 Specification for ArMinorVersion

Name	`ArMinorVersion`		
Description	This parameter provides the minor version of the AUTOSAR specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the current AUTOSAR version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.3 ArPatchVersion

Table 6 Specification for ArPatchVersion

Name	`ArPatchVersion`		
Description	This parameter provides the patch version of the AUTOSAR specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		

(table continues...)

Table 6 (continued) Specification for ArPatchVersion

Default value	As per the current AUTOSAR version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.4 ModuleId

Table 7 Specification for ModuleId

Name	`ModuleId`		
Description	This parameter provides the module ID of DMA		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	255		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.5 Release

Table 8 Specification for Release

Name	`Release`		
Description	This parameter indicates the TC3xx device derivative used for the implementation.		
Multiplicity	1..1	**Type**	EcucStringParamDef
Range	String		
Default value	As per hardware derivative		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

(table continues...)

Table 8 (continued) Specification for Release

Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.6 SwMajorVersion

Table 9 Specification for SwMajorVersion

Name	`SwMajorVersion`		
Description	This parameter provides the major version of the software.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per Driver version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.7 SwMinorVersion

Table 10 Specification for SwMinorVersion

Name	`SwMinorVersion`		
Description	This parameter provides the minor version of the software.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per Driver version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL

(table continues...)

Table 10 (continued) Specification for SwMinorVersion	
Dependency	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.1.1.8 SwPatchVersion

Table 11 Specification for SwPatchVersion			
Name	`SwPatchVersion`		
Description	This parameter provides the patch version of the software.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per Driver version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.9 VendorId

Table 12 Specification for VendorId			
Name	`VendorId`		
Description	This parameter provides the vendor ID		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.2 Container: Dma

Post-Build Variant Multiplicity: FALSE
Multiplicity Configuration Class: -

1.3.1.3 Container: DmaChannelConfig

This container holds the channel wise configuration data for the DMA driver.
Post-Build Variant Multiplicity: FALSE
Multiplicity Configuration Class: Pre-Compile

1.3.1.3.1 DmaChannelAssignedPartition

Table 13 Specification for DmaChannelAssignedPartition

Name	`DmaChannelAssignedPartition`		
Description	This parameter determines the hardware partition which this channel is allocated to. *Note: The default value of this parameter is set to the reset value of the corresponding SFR.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 3		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.2 DmaChannelId

Table 14 Specification for DmaChannelId

Name	`DmaChannelId`		
Description	This parameter determines the physical channel number *Note: The default value is an incremental value starting from zero, to make it unique automatically.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 127		
Default value	Incremental value starting with 0 and unique among all the channels		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-

(table continues...)

Table 14 (continued) Specification for DmaChannelId

Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.3 DmaChannelNotification

Table 15 Specification for DmaChannelNotification

Name	`DmaChannelNotification`		
Description	The DmaChannelNotification is used by the DMA driver to invoke the user-defined function for notification purposes. The parameter can be a name or the address(numeric value) of the notification function. *Note1: By default, the notification parameter will be NULL.* *Note2: The DMA driver does not validate the configured function name or address for correctness and the responsibility falls on the user.* *Note3: If the function names are used for this parameter, instead of address value, then this parameter would be having the value configuration class as link-time.*		
Multiplicity	1..1	**Type**	EcucFunctionNameDef
Range	String		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.4 DmaChannelNumTransactionSet

Table 16 Specification for DmaChannelNumTransactionSet

Name	`DmaChannelNumTransactionSet`

(table continues...)

Table 16 (continued) Specification for DmaChannelNumTransactionSet

Description	This parameter indicates the number of transaction sets which can be configured to the particular channel. Linked list feature is offered in Transaction set configuration only if the value of this parameter is greater than one. The number of transaction sets is capped at DmaMaxTransactionSetPerChannel (to a max 0xFFFF). *Note: Minimum TCS number is selected as the default value.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	1 - DmaMaxTransactionSetPerChannel (capped to a max 0xFFFF)		
Default value	1		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	DmaMaxTransactionSetPerChannel		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.5 DmaChannelUser

Table 17 Specification for DmaChannelUser

Name	`DmaChannelUser`		
Description	This parameter defines the user of the DMA channel. The description field of Tresos tool indicates the different users possible. *Note: The user has to use this parameter in their respective codegen logic to generate the channel IDs which they are using.* *Note: The default value of this parameter is set to - Other.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	CRC: User is CRC library. HSSL: User is HSSL driver. Other: User is any other module not listed. SPI: User is SPI driver.		
Default value	Other		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-

(table continues...)

Table 17 (continued) Specification for DmaChannelUser

Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.6 DmaErrorNotification

Table 18 Specification for DmaErrorNotification

Name	`DmaErrorNotification`		
Description	The DmaErrorNotification is used by the DMA driver to invoke the user-defined function for notification of resource partition error. The parameter can be a name or the address(numeric value) of the notification function. *Note 1: By default, the notification parameter will be NULL.* *Note 2: The DMA driver does not validate the configured function name or address for correctness and the responsibility falls on the user.* *Note3: If the function names are used for this parameter, instead of address value, then this parameter would be having the value configuration class as link-time.*		
Multiplicity	1..1	**Type**	EcucFunctionNameDef
Range	String		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	DmaMESourceErrorInterrupt, DmaMEDestinationErrorInterrupt, DmaMELinkedListErrorInterrupt		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.7 DmaTcsInterruptTransactionLoss

Table 19 Specification for DmaTcsInterruptTransactionLoss

Name	`DmaTcsInterruptTransactionLoss`		
Description	This parameter enables/disables transaction loss interrupt. Notification for the TRL interrupt would be provided only if this TRL bit is enabled. *Note: The optional features are disabled by default to minimize the executable code size.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef

(table continues...)

Table 19	(continued) Specification for DmaTcsInterruptTransactionLoss		
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4 Container: DmaChannelTransactionSet

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: Pre-Compile

1.3.1.4.1 DmaCLLNextIndex

Table 20	Specification for DmaCLLNextIndex		
Name	`DmaCLLNextIndex`		
Description	This parameter defines the next TCS node in case of conditional linked list match feature. *Note: The default value is kept as zero.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - Number of DmaTcsIndex available in Dma channel (max 0xFFFF)		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	DmaTcsShadowRegisterConfiguration, DmaTcsMoveLength		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.2 DmaNextTcsIndex

Table 21	Specification for DmaNextTcsIndex
Name	`DmaNextTcsIndex`

(table continues...)

Table 21 (continued) Specification for DmaNextTcsIndex

Description	This parameter defines the next TCS node in case of linked list feature. *Note: The next transaction set index is selected as the default value of the parameter.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - Number of DmaTcsIndex available in Dma channel (max 0xFFFF)		
Default value	DmaTcsIndex+1 (Incrementing index)		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.3 DmaPatternMode

Table 22 Specification for DmaPatternMode

Name	`DmaPatternMode`		
Description	This parameter defines the mode used during a pattern match. The user should choose the mode as needed for the 8bit, 16bit or 32bit pattern match. The modes are mapped to the values of PATSEL bitfield in CHCFGR register. Please refer hardware user manual for more details. *Note:* *--This configuration parameter will not be used if the value of DmaTcsShadowRegisterConfiguration is DMA_CONDITIONAL_LINKED_LIST.* *--This configuration parameter is valid only for the channels reserved in the DmaPattenConfig container.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	Pattern_Detection_Disabled: Pattern match disabled Pattern_Mode_1: Pattern match mode 1 Pattern_Mode_2: Pattern match mode 2 Pattern_Mode_3: Pattern match mode 3 Pattern_Mode_5: Pattern match mode 5 Pattern_Mode_6: Pattern match mode 6 Pattern_Mode_7: Pattern match mode 7		
Default value	Pattern_Detection_Disabled		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-

(table continues...)

Table 22 (continued) Specification for DmaPatternMode

Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	DmaTcsShadowRegisterConfiguration		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.4 DmaTcsAppendTimeStamp

Table 23 Specification for DmaTcsAppendTimeStamp

Name	`DmaTcsAppendTimeStamp`		
Description	This parameter determines if time stamp is to be appended after the last DMA move of a transaction. *Note: The optional features are disabled by default to minimize the executable code size.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.5 DmaTcsAutoStartEnable

Table 24 Specification for DmaTcsAutoStartEnable

Name	`DmaTcsAutoStartEnable`		
Description	This parameter enables/disables the autostart feature in the case of a linked list *Note: The optional features are kept disabled by default. To be enabled by user as per the need.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		

(table continues...)

Table 24 (continued) Specification for DmaTcsAutoStartEnable

Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	DmaTcsShadowRegisterConfiguration		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.6 DmaTcsCircularBufferDestinationEnable

Table 25 Specification for DmaTcsCircularBufferDestinationEnable

Name	`DmaTcsCircularBufferDestinationEnable`		
Description	This parameter determines if a circular buffering scheme is to be implemented on the destination buffer. *Note: The optional features are disabled by default to minimize the executable code size.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.7 DmaTcsCircularBufferDestinationLength

Table 26 Specification for DmaTcsCircularBufferDestinationLength

Name	`DmaTcsCircularBufferDestinationLength`

(table continues...)

Table 26 (continued) Specification for DmaTcsCircularBufferDestinationLength

Description	This parameter determines how many bits of the destination address of a circular buffer can actually be modified. Remaining address bits stay unchanged. This parameter is grayed out if DmaTcsCircularBufferDestinationEnable is set to FALSE *Note 1: The user has to ensure that the destination address is aligned accordingly (For e.g. if the circular buffer size is chosen to be 16, the address has to be aligned to the 16 byte boundary).* *Note 2: The default value of this parameter is set to the reset value of the corresponding SFR.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	DMA_CIRCULAR_BUFFER_LENGTH_1024BYTE: Circular buffer length of 1024 byte DMA_CIRCULAR_BUFFER_LENGTH_128BYTE: Circular buffer length of 128 byte DMA_CIRCULAR_BUFFER_LENGTH_16384BYTE: Circular buffer length of 16384 byte DMA_CIRCULAR_BUFFER_LENGTH_16BYTE: Circular buffer length of 16 byte DMA_CIRCULAR_BUFFER_LENGTH_1BYTE: Circular buffer length of 1 byte DMA_CIRCULAR_BUFFER_LENGTH_2048BYTE: Circular buffer length of 2048 byte DMA_CIRCULAR_BUFFER_LENGTH_256BYTE: Circular buffer length of 256 byte DMA_CIRCULAR_BUFFER_LENGTH_2BYTE: Circular buffer length of 2 byte DMA_CIRCULAR_BUFFER_LENGTH_32768BYTE: Circular buffer length of 32768 byte DMA_CIRCULAR_BUFFER_LENGTH_32BYTE: Circular buffer length of 32 byte DMA_CIRCULAR_BUFFER_LENGTH_4096BYTE: Circular buffer length of 4096 byte DMA_CIRCULAR_BUFFER_LENGTH_4BYTE: Circular buffer length of 4 byte DMA_CIRCULAR_BUFFER_LENGTH_512BYTE: Circular buffer length of 512 byte DMA_CIRCULAR_BUFFER_LENGTH_64BYTE: Circular buffer length of 64 byte DMA_CIRCULAR_BUFFER_LENGTH_8192BYTE: Circular buffer length of 8192 byte DMA_CIRCULAR_BUFFER_LENGTH_8BYTE: Circular buffer length of 8 byte		
Default value	DMA_CIRCULAR_BUFFER_LENGTH_1BYTE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	DmaTcsCircularBufferDestinationEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.8 DmaTcsCircularBufferSourceEnable

Table 27 Specification for DmaTcsCircularBufferSourceEnable

Name	`DmaTcsCircularBufferSourceEnable`

(table continues...)

Table 27 (continued) Specification for DmaTcsCircularBufferSourceEnable

Description	This parameter determines if a circular buffering scheme is to be implemented on the source buffer. *Note: The optional features are disabled by default to minimize the executable code size.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.9 DmaTcsCircularBufferSourceLength

Table 28 Specification for DmaTcsCircularBufferSourceLength

Name	`DmaTcsCircularBufferSourceLength`		
Description	This parameter determines how many bits of the source address of a circular buffer can actually be modified. Remaining address bits stay unchanged. This parameter is greyed out if DmaTcsCircularBufferSourceEnable is set to FALSE. *Note 1: The user has to ensure that the source address is aligned accordingly (For e.g. if the circular buffer size is chosen to be 16, the address has to be aligned to the 16 byte boundary).* *Note 2: The default value of this parameter is set to the reset value of the corresponding SFR.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef

(table continues...)

Table 28 (continued) Specification for DmaTcsCircularBufferSourceLength

Range	DMA_CIRCULAR_BUFFER_LENGTH_1024BYTE: Circular buffer length of 1024 byte DMA_CIRCULAR_BUFFER_LENGTH_128BYTE: Circular buffer length of 128 byte DMA_CIRCULAR_BUFFER_LENGTH_16384BYTE: Circular buffer length of 16384 byte DMA_CIRCULAR_BUFFER_LENGTH_16BYTE: Circular buffer length of 16 byte DMA_CIRCULAR_BUFFER_LENGTH_1BYTE: Circular buffer length of 1 byte DMA_CIRCULAR_BUFFER_LENGTH_2048BYTE: Circular buffer length of 2048 byte DMA_CIRCULAR_BUFFER_LENGTH_256BYTE: Circular buffer length of 256 byte DMA_CIRCULAR_BUFFER_LENGTH_2BYTE: Circular buffer length of 2 byte DMA_CIRCULAR_BUFFER_LENGTH_32768BYTE: Circular buffer length of 32768 byte DMA_CIRCULAR_BUFFER_LENGTH_32BYTE: Circular buffer length of 32 byte DMA_CIRCULAR_BUFFER_LENGTH_4096BYTE: Circular buffer length of 4096 byte DMA_CIRCULAR_BUFFER_LENGTH_4BYTE: Circular buffer length of 4 byte DMA_CIRCULAR_BUFFER_LENGTH_512BYTE: Circular buffer length of 512 byte DMA_CIRCULAR_BUFFER_LENGTH_64BYTE: Circular buffer length of 64 byte DMA_CIRCULAR_BUFFER_LENGTH_8192BYTE: Circular buffer length of 8192 byte DMA_CIRCULAR_BUFFER_LENGTH_8BYTE: Circular buffer length of 8 byte		
Default value	DMA_CIRCULAR_BUFFER_LENGTH_1BYTE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	DmaTcsCircularBufferSourceEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.10 DmaTcsDaisyChaining

Table 29 Specification for DmaTcsDaisyChaining

Name	`DmaTcsDaisyChaining`		
Description	This parameter determines whether the adjacent lower channel is daisy chained with this channel. If so, after a transaction successfully completes on this channel, an internal hardware trigger is forwarded to the adjacent channel. *Note: The optional features are kept disabled by default.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		

(table continues...)

Table 29 (continued) Specification for DmaTcsDaisyChaining

Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.11 DmaTcsDataTransferInterrupt

Table 30 Specification for DmaTcsDataTransferInterrupt

Name	`DmaTcsDataTransferInterrupt`		
Description	This parameter enables/disables the data transfer/transaction interrupt. *Note: The interrupts are kept disabled by default. To be enabled by user as per the need.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.12 DmaTcsDestinationAddress

Table 31 Specification for DmaTcsDestinationAddress

Name	`DmaTcsDestinationAddress`

(table continues...)

Table 31 (continued) Specification for DmaTcsDestinationAddress

Description	This parameter defines the start address of the destination memory. The address is to be provided either as an address value in hexadecimal or as the pointer to the location (with the ampersand symbol included in the name, if needed). *Note: The code generator would indicate error in the following cases:* *- If the address field is left blank* *- If there are spaces in the name* *- If a numeric value is used which does not end with 'U', indicating unsigned value.* *Other than these checks, there are no other checks performed by the UI or code generator on the validity of the content entered.* *The DMA driver does not validate the configured address for correctness or data alignment and the responsibility falls on the user.* *If double buffering is enabled, this parameter indicates the first of the two buffers.* *Note: Since the address is user configurable, the default value is kept as NULL_PTR.*		
Multiplicity	1..1	**Type**	EcucStringParamDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Link-Time	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.13 DmaTcsDestinationAddressModificationFactor

Table 32 Specification for DmaTcsDestinationAddressModificationFactor

Name	`DmaTcsDestinationAddressModificationFactor`		
Description	This parameter defines how the destination address changes after every move operation. If DmaTcsCircularBufferDestinationLength is 0 and DmaTcsCircularBufferDestinationEnable is TRUE, the UI widget of this parameter is grayed out. The description field of Tresos is used to explain the significance of each of the values is provided. *Note: The default value of this parameter is set to the reset value of the corresponding SFR.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef

(table continues...)

Table 32 (continued) Specification for DmaTcsDestinationAddressModificationFactor			
Range	DMA_FACTOR_128: Address offset is 128 x DmaChBitsPerMove DMA_FACTOR_16: Address offset is 16 x DmaChBitsPerMove DMA_FACTOR_1: Address offset is 1 x DmaChBitsPerMove DMA_FACTOR_2: Address offset is 2 x DmaChBitsPerMove DMA_FACTOR_4: Address offset is 4 x DmaChBitsPerMove DMA_FACTOR_64: Address offset is 64 x DmaChBitsPerMove DMA_FACTOR_8: Address offset is 8 x DmaChBitsPerMove		
Default value	DMA_FACTOR_1		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.14 DmaTcsDestinationAddressMovement

Table 33 Specification for DmaTcsDestinationAddressMovement			
Name	`DmaTcsDestinationAddressMovement`		
Description	This parameter determines if the destination address after every move operation is to be increased or decreased. *Note: The default value is chosen as 'increasing' for the natural progression. User can choose other values as needed.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	DMA_DECREASING: Address decrements DMA_INCREASING: Address increments		
Default value	DMA_INCREASING		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.15 DmaTcsDoubleBuffer

Table 34 Specification for DmaTcsDoubleBuffer

Name	`DmaTcsDoubleBuffer`		
Description	This parameter defines the start address of the double buffer. If double buffering is enabled, this parameter gets enabled and indicates the second buffer.The address is to be provided either as an address value in hexadecimal or as the pointer to the location (with the ampersand symbol included in the name, if needed). With double buffering disabled, this parameter is grayed out. *Note: The code generator would indicate error in the following cases:* *- If the address field is left blank* *- If there are spaces in the name* *- If a numeric value is used which does not end with 'U', indicating unsigned value* *Other than these checks, there are no other checks performed by the UI or code generator on the validity of the content entered.* *The DMA driver does not validate the configured address for correctness or data alignment and the responsibility falls on the user.* *Note: Since the address is user configurable, the default value is kept as NULL_PTR.*		
Multiplicity	1..1	**Type**	EcucStringParamDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Link-Time	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	DmaTcsShadowRegisterConfiguration		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.16 DmaTcsHardwareTrigger

Table 35 Specification for DmaTcsHardwareTrigger

Name	`DmaTcsHardwareTrigger`		
Description	This parameter determines whether hardware initiated trigger is to be supported or not and if affirmative its mode *Note: Since the hardware trigger is user configurable, the default value is kept as disabled.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef

(table continues...)

Table 35 (continued) Specification for DmaTcsHardwareTrigger

Range	DMA_HARDWARE_TRIGGER_CONTINUOUS_MODE: After a DMA transaction, bit TSR.HTRE remains set, ready to accept the next trigger. DMA_HARDWARE_TRIGGER_SINGLE_MODE: After a DMA transaction, the DMA channel is disabled for further hardware requests DMA_NO_HARDWARE_TRIGGER: No hardware trigger would be used		
Default value	DMA_NO_HARDWARE_TRIGGER		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.17 DmaTcsIndex

Table 36 Specification for DmaTcsIndex

Name	`DmaTcsIndex`		
Description	This parameter defines the index of current TCS node in the list of TCS nodes. *Note: The current transaction set index is selected as the default value of the parameter.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - DmaChannelNumTransactionSet -1 (max 0xFFFE)		
Default value	DmaChannelTransactionSet current Index		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.18 DmaTcsInterruptDataTransfer

Table 37 Specification for DmaTcsInterruptDataTransfer

Name	`DmaTcsInterruptDataTransfer`

(table continues...)

Table 37 (continued) Specification for DmaTcsInterruptDataTransfer			
Description	Interrupts can be generated either for every transfer or even transaction. Additionally, an interrupt can be generated after a certain number of transfers equaling a threshold have been completed *Note: The default value of this parameter is set to the reset value of the corresponding SFR.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	DMA_INTERRUPT_AFTER_THRESHOLD: Interrupt is triggered when TCOUNT equals IRDV DMA_INTERRUPT_PER_TRANSACTION: Interrupt is triggered after the transaction is complete. DMA_INTERRUPT_PER_TRANSFER: Interrupt is triggered after the transfer is complete.		
Default value	DMA_INTERRUPT_PER_TRANSACTION		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	DmaTcsDataTransferInterrupt		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.19 DmaTcsInterruptDataTransferThreshold

Table 38 Specification for DmaTcsInterruptDataTransferThreshold			
Name	`DmaTcsInterruptDataTransferThreshold`		
Description	If the parameter DmaTcsInterruptDataTransfer is set to a value of INTERRUPT_AFTER_THRESHOLD, this parameter determines the threshold of transfer following which an interrupt is generated. *Note: The default value of this parameter is set to the reset value of the corresponding SFR.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 15		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	DmaTcsInterruptDataTransfer		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.20 DmaTcsInterruptDestAddressWrap

Table 39 Specification for DmaTcsInterruptDestAddressWrap

Name	`DmaTcsInterruptDestAddressWrap`		
Description	This parameter enables/disables destination address wrap around interrupt. *Note: The interrupts are kept disabled by default. To be enabled by user as per the need.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.21 DmaTcsInterruptSourceAddressWrap

Table 40 Specification for DmaTcsInterruptSourceAddressWrap

Name	`DmaTcsInterruptSourceAddressWrap`		
Description	This parameter enables/disables source address wrap around interrupt *Note: The interrupts are kept disabled by default. To be enabled by user as per the need.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.22 DmaTcsMoveLength

Table 41 Specification for DmaTcsMoveLength

Name	DmaTcsMoveLength		
Description	This parameter defines the amount of data transferred per move operation. The description field of Tresos tool explains the significance of the range of input values. *Note: The channel data width has to be chosen depending on whether the SRI or SPB bus is used. User has to ensure that the appropriate channel width is chosen. Refer the Target Specification for the valid configurations.* *Note: The default value of this parameter is set to the reset value of the corresponding SFR.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	DMA_WIDTH_128BITS: Number of bits per move is 128 bits DMA_WIDTH_16BITS: Number of bits per move is 16 bits DMA_WIDTH_256BITS: Number of bits per move is 256 bits DMA_WIDTH_32BITS: Number of bits per move is 32 bits DMA_WIDTH_64BITS: Number of bits per move is 64 bits DMA_WIDTH_8BITS: Number of bits per move is 8 bits		
Default value	DMA_WIDTH_8BITS		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.23 DmaTcsReferenceAddressCrc

Table 42 Specification for DmaTcsReferenceAddressCrc

Name	DmaTcsReferenceAddressCrc		
Description	This parameter defines the reference address CRC calculated by the user. No checks are performed by the UI or code generator on the validity of the CRC entered. Value is input in hexadecimal format. *Note: Since the CRC value is user configurable, the default value is kept as zero.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0x00000000 - 0xFFFFFFFF		
Default value	0x00000000		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-

(table continues...)

Table 42 (continued) Specification for DmaTcsReferenceAddressCrc

Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	DmaTcsShadowRegisterConfiguration		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.24 DmaTcsReferenceDataCrc

Table 43 Specification for DmaTcsReferenceDataCrc

Name	`DmaTcsReferenceDataCrc`		
Description	This parameter defines the reference data CRC calculated by the user. No checks are performed by the UI or code generator on the validity of the CRC entered. Value is input in hexadecimal format. *Note: Since the CRC value is user configurable, the default value is kept as zero.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0x00000000 - 0xFFFFFFFF		
Default value	0x00000000		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.25 DmaTcsShadowRegisterConfiguration

Table 44 Specification for DmaTcsShadowRegisterConfiguration

Name	`DmaTcsShadowRegisterConfiguration`

(table continues...)

Table 44 (continued) Specification for DmaTcsShadowRegisterConfiguration

Description	This parameter determines how the shadow register is to be used. Details are in the Internal Target Specification. If the parameter DmaChannelNumTransactionSet is set to a value greater than one, user is allowed to choose only the linked list options. Further, it is only for the first node of the linked list that the user is allowed to choose the specific linked list type. All other non-leaf nodes of this linked list inherit the property of the root transaction set. The leaf node of the linked list has SHADOWING_DISABLED configured *Note:* *--The default value of this parameter is set to the reset value of the corresponding SFR.* *-- If the parameter is set to DMA_CONDITIONAL_LINKED_LIST then then DmaPatternMode will not be used.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	DMA_ACCUMULATED_LINKED_LIST: Accumulated Linked List (ACCLL) DMA_CONDITIONAL_LINKED_LIST: Conditional Linked List (CONLL) DMA_DEST_ADDRESS_BUFFERING_RO: Shadow Operation Read Only Mode Destination Address DMA_DEST_ADDRESS_BUFFERING_RW: Shadow Operation Direct Write Mode Destination Address DMA_DEST_DOUBLE_BUFFERING_HW_SW_SWITCH: DMA Double Destination Buffering with Software Switch and Automatic Hardware Switch DMA_DEST_DOUBLE_BUFFERING_SW_SWITCH: DMA Double Destination Buffering with Software Switch Only DMA_LINKED_LIST: DMA Linked List (DMALL) DMA_SAFE_LINKED_LIST: Safe Linked List (SAFLL) DMA_SHADOWING_DISABLED: Shadow operation disabled DMA_SOURCE_ADDRESS_BUFFERING_RO: Shadow Operation Read Only Mode Source Address DMA_SOURCE_ADDRESS_BUFFERING_RW: Shadow Operation Direct Write Mode Source Address DMA_SOURCE_DOUBLE_BUFFERING_HW_SW_SWITCH: DMA Double Source Buffering with Software Switch and Automatic Hardware Switch DMA_SOURCE_DOUBLE_BUFFERING_SW_SWITCH: DMA Double Source Buffering with Software Switch Only		
Default value	DMA_SHADOWING_DISABLED		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

(table continues...)

Table 44	(continued) Specification for DmaTcsShadowRegisterConfiguration
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.1.4.26 DmaTcsSourceAddress

Table 45	Specification for DmaTcsSourceAddress		
Name	`DmaTcsSourceAddress`		
Description	This parameter defines the start address of the source memory. The address is to be provided either as an address value in hexadecimal or as the pointer to the location (with the ampersand symbol included in the name, if needed). *Note: The code generator would indicate error in the following cases:* *- If the address field is left blank* *- If there are spaces in the name* *- If a numeric value is used which does not end with 'U', indicating unsigned value* *Other than these checks, there are no other checks performed by the UI or code generator on the validity of the content entered.* *The DMA driver does not validate the configured address for correctness or data alignment and the responsibility falls on the user.* *If double buffering is enabled, this parameter indicates the first of the two buffers.* *Note: Since the address is user configurable, the default value is kept as NULL_PTR.*		
Multiplicity	1..1	**Type**	EcucStringParamDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Link-Time	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.27 DmaTcsSourceAddressModificationFactor

Table 46	Specification for DmaTcsSourceAddressModificationFactor
Name	`DmaTcsSourceAddressModificationFactor`

(table continues...)

Table 46 (continued) Specification for DmaTcsSourceAddressModificationFactor

Description	This parameter defines how the source address changes after every move operation. If DmaTcsCircularBufferSourceLength is 0 and DmaTcsCircularBufferSourceEnable is TRUE, the UI widget of this parameter is grayed out. The description field of Tresos is used to explain the significance of each of the values is provided. *Note: The default value of this parameter is set to the reset value of the corresponding SFR.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	DMA_FACTOR_128: Address offset is 128 x DmaChBitsPerMove DMA_FACTOR_16: Address offset is 16 x DmaChBitsPerMove DMA_FACTOR_1: Address offset is 1 x DmaChBitsPerMove DMA_FACTOR_2: Address offset is 2 x DmaChBitsPerMove DMA_FACTOR_32: Address offset is 32 x DmaChBitsPerMove DMA_FACTOR_4: Address offset is 4 x DmaChBitsPerMove DMA_FACTOR_64: Address offset is 64 x DmaChBitsPerMove DMA_FACTOR_8: Address offset is 8 x DmaChBitsPerMove		
Default value	DMA_FACTOR_1		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.28 DmaTcsSourceAddressMovement

Table 47 Specification for DmaTcsSourceAddressMovement

Name	`DmaTcsSourceAddressMovement`		
Description	This parameter determines if the source address after every move operation is to be increased or decreased. *Note: The default value is chosen as 'increasing' for the natural progression. User can choose other values as needed.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	DMA_DECREASING: Address decrements DMA_INCREASING: Address increments		
Default value	DMA_INCREASING		

(table continues...)

Table 47 (continued) Specification for DmaTcsSourceAddressMovement

Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.29 DmaTcsSwapDataCRCByteOrder

Table 48 Specification for DmaTcsSwapDataCRCByteOrder

Name	`DmaTcsSwapDataCRCByteOrder`		
Description	This parameter enables/disables the swap CRC byte order feature. *Note: The optional features are kept disabled by default. To be enabled by user as per the need.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.30 DmaTcsTransactionLength

Table 49 Specification for DmaTcsTransactionLength

Name	`DmaTcsTransactionLength`		
Description	This parameter defines how many transfers are performed per DMA transaction. Value 0 and 1 will have same impact (i.e. one transfer) as DMA hardware executes at least one DMA transfer on a channel start. *Note: Since the length value is user configurable, the default value is kept as zero.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef

(table continues...)

Table 49 (continued) Specification for DmaTcsTransactionLength

Range	0 - 16383		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.31 DmaTcsTransferLength

Table 50 Specification for DmaTcsTransferLength

Name	`DmaTcsTransferLength`		
Description	This parameter defines how many moves are performed per DMA transfer. The description field of Tresos tool explains the significance of the range of input values. *Note: The default value of this parameter is set to the reset value of the corresponding SFR.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	DMA_MOVES_16: One transfer has 16 move DMA_MOVES_1: One transfer has 1 move DMA_MOVES_2: One transfer has 2 move DMA_MOVES_3: One transfer has 3 move DMA_MOVES_4: One transfer has 4 move DMA_MOVES_5: One transfer has 5 move DMA_MOVES_8: One transfer has 8 move DMA_MOVES_9: One transfer has 9 move		
Default value	DMA_MOVES_1		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.32 DmaTcsTriggerFrequency

Table 51 Specification for DmaTcsTriggerFrequency

Name	`DmaTcsTriggerFrequency`		
Description	This parameter determines how much of data will the move engine move upon a trigger request (either hardware or software trigger). The move engine, upon receipt of a trigger request, could effectuate a transfer or even a transaction based on the choice made. *Note: The default value of this parameter is set to the reset value of the corresponding SFR.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	DMA_TRANSACTION_PER_TRIGGER: One DMA request starts a complete DMA transaction DMA_TRANSFER_PER_TRIGGER: A DMA request is required for each DMA transfer		
Default value	DMA_TRANSFER_PER_TRIGGER		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.33 DmaUserHeaderFileWithExternDeclarations

Table 52 Specification for DmaUserHeaderFileWithExternDeclarations

Name	`DmaUserHeaderFileWithExternDeclarations`		
Description	This parameter takes the header file which contains the extern declarations of the variable names used to define the source address, destination address or double buffer address. *Note: The default value is kept as NULL. This field gets enabled only when a non-address value is entered for the source address or destination address or double buffer address.*		
Multiplicity	1..1	**Type**	EcucStringParamDef
Range	String		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	DmaTcsDoubleBuffer, DmaTcsDestinationAddress, DmaTcsSourceAddress		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5 Container: DmaGeneral

This container contains the general configuration parameters needed for the DMA driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.5.1 DmaBufferSwitchApi

Table 53 Specification for DmaBufferSwitchApi

Name	`DmaBufferSwitchApi`		
Description	This parameter results in generation of a preprocessor macro that conditionally includes code related to buffer switch interface *Note: The optional APIs are disabled by default to minimize the executable code size.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.2 DmaChDeInitApi

Table 54 Specification for DmaChDeInitApi

Name	`DmaChDeInitApi`		
Description	Adds / removes the service Dma_ChDeInit() from the code. true: Dma_ChDeInit() can be used. false: Dma_ChDeInit() cannot be used. *Note: The optional APIs are disabled by default to minimize the executable code size.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		

(table continues...)

Table 54 (continued) Specification for DmaChDeInitApi

Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.3 DmaDataPendingApi

Table 55 Specification for DmaDataPendingApi

Name	`DmaDataPendingApi`		
Description	Adds / removes the service Dma_ChGetRemainingData() from the code. true: Dma_ChGetRemainingData() can be used. false: Dma_ChGetRemainingData() cannot be used. *Note: The optional APIs are disabled by default to minimize the executable code size.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.4 DmaDeinitApi

Table 56 Specification for DmaDeinitApi

Name	`DmaDeinitApi`
Description	This parameter adds or removes the Dma_DeInit() API from the code. The default value of this parameter is set to FALSE to minimize the executable code size.

(table continues...)

Table 56 (continued) Specification for DmaDeinitApi

Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.5 DmaDevErrorDetect

Table 57 Specification for DmaDevErrorDetect

Name	`DmaDevErrorDetect`		
Description	Switches the Default Error Tracer (DET) detection and notification ON or OFF. true: enabled (ON). false: disabled (OFF).		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.6 DmaGetVersionInfoApi

Table 58 Specification for DmaGetVersionInfoApi

Name	`DmaGetVersionInfoApi`

(table continues...)

Table 58 (continued) Specification for DmaGetVersionInfoApi

Description	Adds / removes the API Dma_GetVersionInfo() from the code. true: Dma_GetVersionInfo() can be used. false: Dma_GetVersionInfo() cannot be used. *Note: The optional APIs are disabled by default to minimize the executable code size.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.7 DmaInitCheckApi

Table 59 Specification for DmaInitCheckApi

Name	`DmaInitCheckApi`		
Description	Adds / removes the service Dma_InitCheck() from the code. true: Dma_InitCheck() can be used. false: Dma_InitCheck() cannot be used. *Note: The optional APIs are disabled by default to minimize the executable code size.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.8 DmaInitDeInitApiMode

Table 60 Specification for DmaInitDeInitApiMode

Name	`DmaInitDeInitApiMode`		
Description	This configuration parameter gives the mode in which the Dma Init and De-Init APIs will be used. *Note: Since DMA driver accesses the SFRs, it is more efficient to operate the DMA driver in supervisor mode. Hence, the default mode of operation is supervisor.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	DMA_MCAL_SUPERVISORMODE: Operating mode used is Supervisor DMA_MCAL_USER1MODE: Operating mode used is USER1		
Default value	DMA_MCAL_SUPERVISORMODE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	DmaRuntimeApiMode		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.9 DmaMaxTransactionSetPerChannel

Table 61 Specification for DmaMaxTransactionSetPerChannel

Name	`DmaMaxTransactionSetPerChannel`		
Description	This parameter controls how many transaction sets are allowed to be defined per channel. No code is generated based on this parameter *Note: Minimum TCS number is selected as the default value.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	1 - 65535		
Default value	1		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.10 DmaMultiCoreErrorDetect

Table 62 Specification for DmaMultiCoreErrorDetect

Name	`DmaMultiCoreErrorDetect`		
Description	The parameter enables or disables the multi core related default error tracer (DET) detection and reporting. It is applicable only when DET/Safety is enabled. *Note: This parameter should be kept false for single core systems.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	DmaDevErrorDetect		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.11 DmaRuntimeApiMode

Table 63 Specification for DmaRuntimeApiMode

Name	`DmaRuntimeApiMode`		
Description	This configuration parameter gives the mode in which the Runtime API will be used. *Note: Since DMA driver accesses the SFRs, it is more efficient to operate the DMA driver in supervisor mode. Hence, the default mode of operation is supervisor.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	DMA_MCAL_SUPERVISORMODE: Operating mode used is Supervisor DMA_MCAL_USER1MODE: Operating mode used is USER1		
Default value	DMA_MCAL_SUPERVISORMODE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.12 DmaSafetyEnable

Table 64 Specification for DmaSafetyEnable

Name	`DmaSafetyEnable`		
Description	Switch to enable/disable the safety check and reporting. true: Enable safety check and reporting false: Disable safety check and reporting *Note: The detection of safety related errors is enabled by default to ensure that safety issues are addressed during the product lifecycle.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.13 DmaSuspendApi

Table 65 Specification for DmaSuspendApi

Name	`DmaSuspendApi`		
Description	Adds / removes the services Dma_ChTransferFreeze() and Dma_ChTransferResume() from the code. true: Dma_ChTransferFreeze() and Dma_ChTransferResume() can be used. false: Dma_ChTransferFreeze() and Dma_ChTransferResume() cannot be used. *Note: The optional APIs are disabled by default to minimize the executable code size.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

(table continues...)

Table 65 (continued) Specification for DmaSuspendApi

Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.14 DmaTriggerApi

Table 66 Specification for DmaTriggerApi

Name	`DmaTriggerApi`		
Description	Adds / removes the services Dma_ChEnableHardwareTrigger() and Dma_ChDisableHardwareTrigger() from the code. true: Dma_ChEnableHardwareTrigger() and Dma_ChDisableHardwareTrigger() can be used. false: Dma_ChEnableHardwareTrigger() and Dma_ChDisableHardwareTrigger() cannot be used. *Note: The optional APIs are disabled by default to minimize the executable code size.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6 Container: DmaMoveEngineConfig

DmaMoveEngineConfig contains the configuration parameters for the notifications from the move engine.
Post-Build Variant Multiplicity: -
Multiplicity Configuration Class: -

1.3.1.6.1 DmaMEDestinationErrorInterrupt

Table 67 Specification for DmaMEDestinationErrorInterrupt

Name	`DmaMEDestinationErrorInterrupt`		
Description	This parameter enables/disables Move engines destination error interrupt. *Note: The interrupts are kept disabled by default. To be enabled by user as per the need.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.2 DmaMELinkedListErrorInterrupt

Table 68 Specification for DmaMELinkedListErrorInterrupt

Name	`DmaMELinkedListErrorInterrupt`		
Description	This parameter enables/disables linked list error interrupts *Note: The interrupts are kept disabled by default. To be enabled by user as per the need.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.3 DmaMESourceErrorInterrupt

Table 69 Specification for DmaMESourceErrorInterrupt

Name	`DmaMESourceErrorInterrupt`		
Description	This parameter enables/disables Move engines source error interrupt. *Note: The interrupts are kept disabled by default. To be enabled by user as per the need.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7 Container: DmaPatternConfig

This container contains the configuration parameters for the pattern matching feature.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.7.1 DmaChannelForPattern0

Table 70 Specification for DmaChannelForPattern0

Name	`DmaChannelForPattern0`

(table continues...)

Table 70 (continued) Specification for DmaChannelForPattern0

Description	This parameter is to set the channel which uses the PRR0 register. *Note:* *- The default value of this parameter is set to NULL as it has to be configured by the user.* *-DmaChannelForPattern0 and DmaChannelForPattern1 should not be equal* *-This parameter is editable if any of the following is satisfied:* *--- For any of the DMA channel, if DmaChannelTransactionSet contains DmaPatternMode with value other than Pattern_Detection_Disabled.* *--- For any of the DMA channel, if the value of DmaTcsShadowRegisterConfiguration is DMA_CONDITIONAL_LINKED_LIST."* Hint: To use Pattern match feature or Conditional Linked list, please click on the ﷺRed iconﷺ present next to the parameter in the Tresos GUI tool to enable and configure the reference value. If Pattern match feature or Conditional Linked list is not used, please click the same 'Green icon' to disable and avoid code generation warning. *Note:* *--Warning cannot be suppressed in the AUTOSAR format BMD file for non-Tresos tool user and can be ignored.* *--ﷺRed iconﷺ changes to 'Green icon' when parameter changes from Disable to Enable and vice versa.*		
Multiplicity	0..1	**Type**	EcucReferenceDef
Range	Reference to Node: DmaChannelConfig		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	DmaTcsShadowRegisterConfiguration, DmaPatternMode		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.2 DmaChannelForPattern1

Table 71 Specification for DmaChannelForPattern1

Name	`DmaChannelForPattern1`

(table continues...)

Table 71 (continued) Specification for DmaChannelForPattern1

Description	This parameter is to set the channel which uses the PRR1 register. *Note:* *-The default value of this parameter is set to NULL as it has to be configured by the user.* *-DmaChannelForPattern1 and DmaChannelForPattern0 should not be equal* *-This parameter is editable if any of the following is satisfied:* *--- For any of the DMA channel, if DmaChannelTransactionSet contains DmaPatternMode with value other than Pattern_Detection_Disabled.* *--- For any of the DMA channel, if the value of DmaTcsShadowRegisterConfiguration is DMA_CONDITIONAL_LINKED_LIST."* Hint: To use Pattern match feature or Conditional Linked list, please click on the ﷺRed iconﷺ present next to the parameter in the Tresos GUI tool to enable and configure the reference value. If Pattern match feature or Conditional Linked list is not used, please click the same 'Green icon' to disable and avoid code generation warning. *Note:* *--Warning cannot be suppressed in the AUTOSAR format BMD file for non-Tresos tool user and can be ignored.* *--ﷺRed iconﷺ changes to 'Green icon' when parameter changes from Disable to Enable and vice versa.*		
Multiplicity	0..1	**Type**	EcucReferenceDef
Range	Reference to Node: DmaChannelConfig		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	DmaTcsShadowRegisterConfiguration, DmaPatternMode		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.3 DmaPattern0

Table 72 Specification for DmaPattern0

Name	`DmaPattern0`		
Description	This parameter is to set the pattern in the PRR0 register. *Note: The default value of this parameter is set to the reset value of the corresponding SFR.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef

(table continues...)

Table 72 (continued) Specification for DmaPattern0

Range	0 - 0xFFFFFFFF		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	DmaChannelForPattern0		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.4 DmaPattern1

Table 73 Specification for DmaPattern1

Name	`DmaPattern1`		
Description	This parameter is to set the pattern in the PRR1 register. *Note: The default value of this parameter is set to the reset value of the corresponding SFR.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 0xFFFFFFFF		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	DmaChannelForPattern1		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8 Container: DmaResourcePartition

DMAResourcePartition contains the configuration related to the resource partitions of DMA.
Post-Build Variant Multiplicity: FALSE
Multiplicity Configuration Class: Pre-Compile

1.3.1.8.1 DmaPermittedBusMaster

Table 74 Specification for DmaPermittedBusMaster

Name	`DmaPermittedBusMaster`

(table continues...)

Table 74 (continued) Specification for DmaPermittedBusMaster

Description	This parameter is a 32 bit hexadecimal mask and represents the bus masters that are allowed to access this partition. *Note: The default value of this parameter is set to the reset value of the corresponding SFR.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0x00000000 - 0xFFFFFFFF		
Default value	0xFFFFFFFF		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.2 DmaResourcePartitionBusMode

Table 75 Specification for DmaResourcePartitionBusMode

Name	`DmaResourcePartitionBusMode`		
Description	This parameter defines the bus access mode of the resource partition *Note: The default value of this parameter is set to the reset value of the corresponding SFR.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	DMA_RP_SUPERVISOR_MODE: Resource partition would be in the supervisor mode. DMA_RP_USER_MODE: Resource partition would be in the user mode.		
Default value	DMA_RP_SUPERVISOR_MODE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.2 Functions - Type definitions

1.3.2.1 Dma_ChannelNotificationPtrType

Table 76 Specification for Dma_ChannelNotificationPtrType

Syntax	`Dma_ChannelNotificationPtrType`
Type	Pointer to a function of type void Function_Name (const uint8 Channel, const uint32 Event)
File	`Dma.h`
Description	This function pointer type would hold the address of the function which has to be invoked when there is a channel event.
Source	IFX
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.2.2 Dma_ConfigType

Table 77 Specification for Dma_ConfigType

Syntax	`Dma_ConfigType`	
Type	Structure	
File	`Dma.h`	
Range	--	None
Description	Defines the type for data structure containing the set of configuration parameters required for initializing the DMA driver and DMA hardware unit(s).	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.3 Dma_ConfigUpdateType

Table 78 Specification for Dma_ConfigUpdateType

Syntax	`Dma_ConfigUpdateType`	
Type	Structure	
File	`Dma.h`	
Range	uint32 SourceAddress	Start of the source buffer address
	uint32 DestAddress	Start of the destination buffer address
	uint32 ControlAdicr	Contents of ADICR register
	uint32 ControlChcsr	Contents of CHCSR register
	uint32 Config	Contents of CHCFGR register
	uint32 ShadowConfig	Contents of SHADR register

(table continues...)

Table 78 (continued) Specification for Dma_ConfigUpdateType

	uint32 AddressCrc	Contents of SDCRC register
	uint32 DataCrc	Contents of RDCRC register
	unsigned_int UpdateSourceAddress:1	Indicates whether the Source address should be updated or not
	unsigned_int UpdateDestAddress:1	Indicates whether the Destination address should be updated or not
	unsigned_int UpdateControlAdicr:1	Indicates whether the ADICR should be updated or not
	unsigned_int UpdateControlChcsr:1	Indicates whether the CHCSR register should be updated or not
	unsigned_int UpdateConfig:1	Indicates whether the CHCFG should be updated or not
	unsigned_int UpdateShadowConfig:1	Indicates whether the SHADR should be updated or not
	unsigned_int UpdateAddressCrc:1	Indicates whether the SDCRC should be updated or not
	unsigned int UpdateDataCrc:1	Indicates whether the RDCRC should be updated or not
Description	Dma_ConfigUpdateType contains the elements which are needed in updating the channel settings of an already configured channel.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.4 Dma_CrcType

Table 79 Specification for Dma_CrcType

Syntax	`Dma_CrcType`	
Type	Enumeration	
File	`Dma.h`	
Range	0 - DMA_NO_CRC_TYPE	Invalid option
	1 - DMA_DATA_CRC_TYPE	Refers to Data CRC
	2 - DMA_ADDRESS_CRC_TYPE	Refers to Address CRC
Description	This enum holds the list of the CRC types available	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.5 Dma_ErrorNotificationPtrType

Table 80 Specification for Dma_ErrorNotificationPtrType

Syntax	`Dma_ErrorNotificationPtrType`
Type	Pointer to a function of type void Function_Name (const uint8 Channel, const uint32 Event)
File	`Dma.h`
Description	This function pointer holds the function which needs to be invoked if there an error event from the move engine. There would be 1 interrupt per resource partition.
Source	IFX
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.2.6 Dma_EventsType

Table 81 Specification for Dma_EventsType

Syntax	`Dma_EventsType`	
Type	Enumeration	
File	`Dma.h`	
Range	0 - DMA_EVENT_NONE	No channel events
	1 - DMA_EVENT_CH_RUNNING	Channel is active and running
	2 - DMA_EVENT_CH_TRANSFER_COMPLETE	Desired transfers completed
	4 - DMA_EVENT_CH_BUFFER_WRAP_SOURCE	Source circular buffer wrap detected
	8 - DMA_EVENT_CH_BUFFER_WRAP_DEST	Destination circular buffer wrap detected
	16 - DMA_EVENT_CH_UNKNOWN_EVENT	Unknown event due to hardware limitations
	32 - DMA_EVENT_CH_TRL_ERROR	Transaction request lost error
	64 - DMA_EVENT_ME_SOURCE_ERROR	Bus error while moving data from source
	128 - DMA_EVENT_ME_DESTINATION_ERROR	Bus error while moving data to destination
	256 - DMA_EVENT_ME_SPB_ERROR	Bus error-SPB
	512 - DMA_EVENT_ME_SRI_ERROR	Bus error-SRI
	1028 - DMA_EVENT_ME_RAM_ERROR	DMARAM access error
	2048 - DMA_EVENT_ME_SAFE_LINKEDLIST_ERROR	CRC comparison error in safe linked list
	4096 - DMA_EVENT_ME_DMA_LINKEDLIST_ERROR	DMARAM access error (during over write of TCS in linked list scenario)
Description	Dma_EventsType is an enumeration holding the list of DMA events, a no-event scenario and an active channel condition.	

(table continues...)

Table 81 (continued) Specification for Dma_EventsType

Source	IFX
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.2.7 Dma_MoveEngineListType

Table 82 Specification for Dma_MoveEngineListType

Syntax	`Dma_MoveEngineListType`	
Type	Enumeration	
File	`Dma.h`	
Range	0 - DMA_ME_NONE	Invalid option
	1 - DMA_ME_0	Refers to Move Engine 0
	2 - DMA_ME_1	Refers to Move Engine 1
	4 - DMA_ME_ALL	Refers to both Move Engines
Description	This enum holds the list of the Move Engines available	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3 Functions - APIs

This section lists all the APIs of the DMA driver.

1.3.3.1 Dma_DeInit

Table 83 Specification for `Dma_DeInit` API

Syntax	`void Dma_DeInit ( void )`	
Service ID	0x18	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	None

(table continues...)

Table 83 (continued) Specification for `Dma_DeInit` API

Description	This API resets all SFRs of the DMA configured during initialization to their reset states. This API must be invoked from all the cores using the DMA driver, as each call resets only the SFRs and global variables of the DMA resources assigned to the calling core. The SFRs and global variables common to all cores are reset by the MCAL master core. This API is only available when DmaDeInitApi is configured as TRUE. *Note: The slave cores should be deinitialized before the master core deinitialization.* *Note: SFRs which are not configured during Dma_Init are not de-initialized by this API.*
Source	IFX
Error handling	DMA_E_DATA_TRANSFER_IN_PROGRESS, DMA_E_SLAVE_INIT
Configuration dependencies	DmaDeinitApi
User hints	None
SFR accessed	CPU_COMPAT(w), CPU_CORE_ID(r), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), DMA_ACCEN_ACCENR0(w), DMA_CH_ADICR(rw), DMA_CH_CHCFGR(w), DMA_CH_CHCSR(w), DMA_CH_DADR(w), DMA_CH_RDCRCR(w), DMA_CH_SADR(w), DMA_CH_SDCRCR(w), DMA_CH_SHADR(w), DMA_CLC(rw), DMA_HRR(w), DMA_ME_CLRE(w), DMA_ME_EER(w), DMA_MODE(w), DMA_PRR0(w), DMA_PRR1(w), DMA_TSR(rw), SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.2 Dma_Init

Table 84 Specification for `Dma_Init` API

Syntax	`void Dma_Init`
	`(`
	`  const Dma_ConfigType * const ConfigPtr`
	`)`	

Service ID	0x01	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to root data structure of all post build configurations
Parameters (out)	-	-

(table continues...)

Table 84 (continued) Specification for `Dma_Init` API

Parameters (in - out)	-	-
Return	void	-
Description	This interface is meant to be called by the ECU initialization routine. The move engines, the resource partitions and the channels are initialized. Note: The master core initialization has to be performed before the slave core initializations.	
Source	IFX	
Error handling	DMA_E_ALREADY_INITIALIZED, DMA_E_MASTER_UNINIT, DMA_E_CORE_NOT_CONFIGURED, DMA_E_NULL_POINTER	
Configuration dependencies	-	
User hints	-	
SFR accessed	CPU_COMPAT(w), CPU_CORE_ID(r), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), DMA_ACCEN_ACCENR0(rw), DMA_CH_ADICR(rw), DMA_CH_CHCFGR(w), DMA_CH_CHCSR(w), DMA_CH_DADR(w), DMA_CH_RDCRCR(w), DMA_CH_SADR(w), DMA_CH_SDCRCR(w), DMA_CH_SHADR(rw), DMA_CLC(rw), DMA_HRR(rw), DMA_ME_CLRE(w), DMA_ME_EER(w), DMA_MODE(rw), DMA_PRR0(w), DMA_PRR1(w), DMA_TSR(w), SCU_CCUCON0(r), SCU_EICON0(rw), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.3 Dma_IsInitDone

Table 85 Specification for `Dma_IsInitDone` API

Syntax	`Std_ReturnType Dma_IsInitDone ( void )`	
Service ID	0x02	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-

(table continues...)

Table 85 (continued) Specification for `Dma_IsInitDone` API

Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK if the driver has been initialized, E_NOT_OK: In Case of - Driver is not initialized
Description	This interface is meant to provide the status whether the initialization process for the DMA driver has been executed or not.	
Source	IFX	
Error handling	-	
Configuration dependencies	-	
User hints	Note: Dma_InitDone API only indicates if the module initialization was invoked or not. In case channels were stopped, the user shall not rely or use this API to derive the initialization status of the channel.	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.4 Dma_ChInit

Table 86 Specification for `Dma_ChInit` API

Syntax	`void Dma_ChInit`
	`(`
	`  const uint8 Channel`
	`)`	

Service ID	0x03	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-

(table continues...)

Table 86 (continued) Specification for `Dma_ChInit` API

Description	This interface is meant to be called to initialize the specified DMA channel with parameters defined in DMAChannelTransactionSet container. This interface is called by the owner (e.g. ADC driver) of the specified channel. A typical use case would be that an ongoing transfer was aborted by invocation of Dma_ChStopTransfer entailing re-initialization of the channel. Any previously triggered interrupt statues would be cleared when the channel init is done.
Source	IFX
Error handling	DMA_E_DRIVER_NOT_INITIALIZED, DMA_E_CHANNEL_INVALID_ID, DMA_E_CH_ALREADY_INITIALIZED, DMA_E_CORE_CHANNEL_MISMATCH
Configuration dependencies	-
User hints	Hint: If a DMA channel is reset (for e.g. due to invocation of 'Dma_ChStopTransfer' or 'Dma_ChDeInit' APIs), the user has to invoke 'Dma_ChInit' to restore the initialization status of the channel.
SFR accessed	CPU_CORE_ID(r), DMA_CH_ADICR(rw), DMA_CH_CHCFGR(w), DMA_CH_CHCSR(w), DMA_CH_DADR(w), DMA_CH_RDCRCR(w), DMA_CH_SADR(w), DMA_CH_SDCRCR(w), DMA_CH_SHADR(w), DMA_HRR(w), DMA_ME_CLRE(w), DMA_ME_ERRSR(r), DMA_TSR(w) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.5 Dma_ChUpdate

Table 87 Specification for `Dma_ChUpdate` API

Syntax	`void Dma_ChUpdate ( const uint8 Channel, const Dma_ConfigUpdateType * const Config, const uint32 * const NodeAddress )`	
Service ID	0x04	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel Config NodeAddress	DMA channel number Pointer to configuration data Start of a memory block which shall be formatted with information in “Config” object
Parameters (out)	-	-

(table continues...)

Table 87 (continued) Specification for `Dma_ChUpdate` API

Parameters (in - out)	-	-
Return	void	-
Description	In certain cases, applications wishing to use the DMA are informed of transfer attributes only at run time. In such cases, it is not possible to use the post-build configuration data. This interface may be used to program transfer characteristics at run time.	
Source	IFX	
Error handling	DMA_E_DRIVER_NOT_INITIALIZED, DMA_E_CHANNEL_INVALID_ID, DMA_E_INVALID_SHADOW_CONFIG_REQ, DMA_E_CORE_CHANNEL_MISMATCH, DMA_E_DATA_TRANSFER_IN_PROGRESS, DMA_E_CHANNEL_INVALID_START_REQ, DMA_E_NULL_POINTER	
Configuration dependencies	-	
User hints	Note: Dma_ChUpdate API should not be called by the user with invalid TCS contents (reserved bit fields selected, unused features selected, read-only or status bits set).	
SFR accessed	CPU_CORE_ID(r), DMA_CH_ADICR(rw), DMA_CH_CHCFGR(w), DMA_CH_CHCSR(w), DMA_CH_DADR(w), DMA_CH_RDCRCR(w), DMA_CH_SADR(w), DMA_CH_SDCRCR(w), DMA_CH_SHADR(rw), DMA_TSR(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.6 Dma_ChDeInit

Table 88 Specification for `Dma_ChDeInit` API

Syntax	`void  Dma_ChDeInit`
	`(`
	`  const uint8 Channel`
	`)`	

Service ID	0x05	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
Parameters (out)	-	-
Parameters (in - out)	-	-

(table continues...)

Table 88 (continued) Specification for `Dma_ChDeInit` API

Return	void	-
Description	This interface is meant to be called to de-initialize a previously configured channel.	
Source	IFX	
Error handling	DMA_E_DRIVER_NOT_INITIALIZED, DMA_E_CHANNEL_INVALID_ID, DMA_E_CHANNEL_NOT_INITIALIZED, DMA_E_CORE_CHANNEL_MISMATCH, DMA_E_DATA_TRANSFER_IN_PROGRESS	
Configuration dependencies	DmaChDeInitApi	
User hints	-	
SFR accessed	CPU_COMPAT(w), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), DMA_CH_ADICR(rw), DMA_CH_CHCFGR(w), DMA_CH_CHCSR(w), DMA_CH_DADR(w), DMA_CH_RDCRCR(w), DMA_CH_SADR(w), DMA_CH_SDCRCR(w), DMA_CH_SHADR(w), DMA_HRR(w), DMA_TSR(rw), SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.7 Dma_ChTransferFreeze

Table 89 Specification for `Dma_ChTransferFreeze` API

Syntax	`void Dma_ChTransferFreeze ( const uint8 Channel )`	
Service ID	0x06	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-

(table continues...)

Table 89 (continued) Specification for `Dma_ChTransferFreeze` API

Description	This interface may be used to temporarily freeze an ongoing transfer. This is typically useful in situations where a block of memory which is an endpoint in a DMA transaction is required to undergo memory testing by a diagnostics task scheduled periodically.
Source	IFX
Error handling	DMA_E_DRIVER_NOT_INITIALIZED, DMA_E_NO_TRANSFERS_PENDING, DMA_E_TIMEOUT, DMA_E_CHANNEL_NOT_INITIALIZED, DMA_E_CHANNEL_INVALID_ID, DMA_E_CORE_CHANNEL_MISMATCH
Configuration dependencies	DmaSuspendApi
User hints	-
SFR accessed	DMA_CH_CHCSR(r), DMA_TSR(rw) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.8 Dma_ChTransferResume

Table 90 Specification for `Dma_ChTransferResume` API

Syntax	`void Dma_ChTransferResume ( const uint8 Channel )`	
Service ID	0x07	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This interface may be used to resume a previously frozen channel.	
Source	IFX	
Error handling	DMA_E_DRIVER_NOT_INITIALIZED, DMA_E_NOT_IN_FREEZE_STATE, DMA_E_CHANNEL_NOT_INITIALIZED, DMA_E_CHANNEL_INVALID_ID, DMA_E_CORE_CHANNEL_MISMATCH	

(table continues...)

Table 90 (continued) Specification for `Dma_ChTransferResume` API

Configuration dependencies	DmaSuspendApi
User hints	The resume interface would put the channel from the freeze (halt) state to the idle state. To continue any DMA operations, which was in progress, the user has to call the start transfer interface.
SFR accessed	DMA_TSR(rw) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.9 Dma_ChEnableHardwareTrigger

Table 91 Specification for `Dma_ChEnableHardwareTrigger` API

Syntax	`void Dma_ChEnableHardwareTrigger ( const uint8 Channel )`	
Service ID	0x08	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This interface is meant to be used to enable (or re-enable) hardware trigger recognition capability. This is particularly important if the channel has been configured for a DMA trigger to be asserted upon detection of a hardware trigger from a peripheral. In “Hardware Trigger-Single” mode of operation, the hardware trigger detection capability is lost after a transaction and shall be reinstalled. This API is precisely meant to do just that.	
Source	IFX	
Error handling	DMA_E_DRIVER_NOT_INITIALIZED, DMA_E_CHANNEL_NOT_INITIALIZED, DMA_E_CHANNEL_INVALID_ID, DMA_E_CORE_CHANNEL_MISMATCH, DMA_E_FREEZE_STATE, DMA_E_DATA_TRANSFER_IN_PROGRESS	
Configuration dependencies	DmaTriggerApi	

(table continues...)

Table 91 (continued) Specification for `Dma_ChEnableHardwareTrigger` API

User hints	Note: The Dma_ChEnableHardwareTrigger API should not be called for a channel which is configured as 'no hardware trigger'
SFR accessed	DMA_TSR(rw) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.10 Dma_ChDisableHardwareTrigger

Table 92 Specification for `Dma_ChDisableHardwareTrigger` API

Syntax	`void Dma_ChDisableHardwareTrigger ( const uint8 Channel )`	
Service ID	0x09	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This interface disables hardware trigger detection capability.	
Source	IFX	
Error handling	DMA_E_DRIVER_NOT_INITIALIZED, DMA_E_CHANNEL_INVALID_ID, DMA_E_CHANNEL_NOT_INITIALIZED, DMA_E_CORE_CHANNEL_MISMATCH, DMA_E_FREEZE_STATE, DMA_E_DATA_TRANSFER_IN_PROGRESS	
Configuration dependencies	DmaTriggerApi	
User hints	Note: The Dma_ChDisableHardwareTrigger API should not be called for a channel which is configured as 'no hardware trigger'	
SFR accessed	DMA_TSR(rw) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	

(table continues...)

Table 92 (continued) Specification for `Dma_ChDisableHardwareTrigger` API	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.11 Dma_ChStartTransfer

Table 93 Specification for `Dma_ChStartTransfer` API		
Syntax	`void Dma_ChStartTransfer ( const uint8 Channel )`	
Service ID	0x0A	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This interface may be used by users of DMA channels (e.g. SPI driver) to manually start a DMA transfer.	
Source	IFX	
Error handling	DMA_E_DRIVER_NOT_INITIALIZED, DMA_E_CHANNEL_NOT_INITIALIZED, DMA_E_CHANNEL_INVALID_ID, DMA_E_FREEZE_STATE, DMA_E_DATA_TRANSFER_IN_PROGRESS, DMA_E_CORE_CHANNEL_MISMATCH	
Configuration dependencies	-	
User hints	-	
SFR accessed	DMA_CH_CHCSR(w), DMA_TSR(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.12 Dma_ChStopTransfer

Table 94 Specification for `Dma_ChStopTransfer` API

Syntax	`void Dma_ChStopTransfer ( const uint8 Channel )`	
Service ID	0x0B	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This interface may be used by users of DMA channels (e.g. ADC driver) to abort an ongoing transfer.	
Source	IFX	
Error handling	DMA_E_DRIVER_NOT_INITIALIZED, DMA_E_CHANNEL_NOT_INITIALIZED, DMA_E_CHANNEL_INVALID_ID, DMA_E_TIMEOUT, DMA_E_CORE_CHANNEL_MISMATCH	
Configuration dependencies	-	
User hints	The channel stoppage would result in the hardware resetting the registers of the channel being used. To use the channel again, the user should call the 'Dma_ChInit' API, to get the registers re-initialized.	
SFR accessed	DMA_TSR(rw) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.13 Dma_ChGetRemainingData

Table 95 Specification for `Dma_ChGetRemainingData` API

Syntax	`uint32 Dma_ChGetRemainingData ( const uint8 Channel )`

(table continues...)

Table 95 (continued) Specification for `Dma_ChGetRemainingData` API

Service ID	0x0C	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	Pending number of Dma transfer for the Dma channel. *Note: If the channel is active (transfers are ongoing) or if there are DET/safety errors present, the API will return the value 0xFFFFFFFF. If 0xFFFFFFFF is returned by the API, the caller should check for the presence of any DET/Safety error.*
Description	This interface may be used by users of DMA channels to find out how many DMA transfers are pending to copy data from source to destination address.	
Source	IFX	
Error handling	DMA_E_DRIVER_NOT_INITIALIZED, DMA_E_CHANNEL_NOT_INITIALIZED, DMA_E_CHANNEL_INVALID_ID, DMA_E_CORE_CHANNEL_MISMATCH, DMA_E_DATA_TRANSFER_IN_PROGRESS	
Configuration dependencies	DmaDataPendingApi	
User hints	-	
SFR accessed	DMA_CH_CHCSR(r), DMA_TSR(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.14 Dma_ChSwitchBuffer

Table 96 Specification for `Dma_ChSwitchBuffer` API

Syntax	`void Dma_ChSwitchBuffer ( const uint8 Channel )`
Service ID	0x0D
Sync/Async	Synchronous

(table continues...)

Table 96 (continued) Specification for `Dma_ChSwitchBuffer` API

Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This interface may be used in a double buffering scheme, to redirect the data from the source to destination in either of the following ways: 1. From two source buffers to a destination buffer or 2. From a source buffer to two destination buffers.	
Source	IFX	
Error handling	DMA_E_DRIVER_NOT_INITIALIZED, DMA_E_CHANNEL_INVALID_ID, DMA_E_CHANNEL_NOT_INITIALIZED, DMA_E_CHANNEL_INVALID_SWITCH_REQ, DMA_E_CORE_CHANNEL_MISMATCH, DMA_E_DATA_TRANSFER_IN_PROGRESS	
Configuration dependencies	DmaBufferSwitchApi	
User hints	-	
SFR accessed	DMA_CH_ADICR(r), DMA_CH_CHCSR(w), DMA_TSR(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.15 Dma_GetEvents

Table 97 Specification for `Dma_GetEvents` API

Syntax	`uint32  Dma_GetEvents`
	`(`
	`  const uint8 Channel`
	`)`	

Service ID	0x0E	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number

(table continues...)

Table 97 (continued) Specification for `Dma_GetEvents` API

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	The 32bit status word containing the consolidated channel events. *Note 1: If the DMA channel is active, the API would return 'DMA_EVENT_CH_RUNNING'. In that case, the caller should call the API again to get the events information from DMA.* *Note 2: The API would return zero in case: (1) If there are any DET or safety errors present or (2) If there are no events recorded in the SFRs. When the return value is zero, the caller should check for the presence of DET or safety errors.*
Description	This interface may be used by users of DMA channels to get all the events which occurred for the DMA channel. The events include the channel events and the move engine events.	
Source	IFX	
Error handling	DMA_E_DRIVER_NOT_INITIALIZED, DMA_E_CHANNEL_NOT_INITIALIZED, DMA_E_CORE_CHANNEL_MISMATCH, DMA_E_CHANNEL_INVALID_ID	
Configuration dependencies	-	
User hints	-	
SFR accessed	DMA_CH_CHCSR(r), DMA_ME_ERRSR(r), DMA_TSR(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.16 Dma_ChStatusClear

Table 98 Specification for `Dma_ChStatusClear` API

Syntax	`void Dma_ChStatusClear`
	`(`
	`  const uint8 Channel,`
	`  const Dma_EventsType Event`
	`)`

Service ID	0x0F
Sync/Async	Synchronous
Safety Level	Refer to the release notes for the safety related info
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels

(table continues...)

Table 98 (continued) Specification for `Dma_ChStatusClear` API

Parameters (in)	Channel Event	DMA channel number Event whose status is to be cleared
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This interface may be used by users of DMA channels (e.g. SPI driver) to reset the specified status event.	
Source	IFX	
Error handling	DMA_E_DRIVER_NOT_INITIALIZED, DMA_E_CHANNEL_NOT_INITIALIZED, DMA_E_CHANNEL_INVALID_EVENT, DMA_E_CHANNEL_INVALID_ID, DMA_E_CORE_CHANNEL_MISMATCH, DMA_E_DATA_TRANSFER_IN_PROGRESS	
Configuration dependencies	-	
User hints	-	
SFR accessed	DMA_CH_CHCSR(w), DMA_TSR(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.17 Dma_ChInterruptEnable

Table 99 Specification for `Dma_ChInterruptEnable` API

Syntax	`void Dma_ChInterruptEnable`
	`(`
	`  const uint8 Channel,`
	`  const Dma_EventsType Event`
	`)`	

Service ID	0x10	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel Event	DMA channel number Event whose status is to be cleared
Parameters (out)	-	-

(table continues...)

Table 99 (continued) Specification for `Dma_ChInterruptEnable` API

Parameters (in - out)	-	-
Return	void	-
Description	This interface may be used by users of DMA channels (e.g. SPI driver) to enable interrupt generation upon occurrence of the specified event.	
Source	IFX	
Error handling	DMA_E_DRIVER_NOT_INITIALIZED, DMA_E_CHANNEL_NOT_INITIALIZED, DMA_E_CHANNEL_INVALID_EVENT, DMA_E_CHANNEL_INVALID_ID, DMA_E_CORE_CHANNEL_MISMATCH, DMA_E_DATA_TRANSFER_IN_PROGRESS, DMA_E_CH_NO_NOTIFICATION_CONFIGURED, DMA_E_FREEZE_STATE	
Configuration dependencies	-	
User hints	-	
SFR accessed	CPU_CORE_ID(r), DMA_CH_ADICR(w), DMA_CH_CHCSR(w), DMA_TSR(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.18 Dma_ChInterruptDisable

Table 100 Specification for `Dma_ChInterruptDisable` API

Syntax	`void Dma_ChInterruptDisable ( const uint8 Channel, const Dma_EventsType Event )`	
Service ID	0x11	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel Event	DMA channel number Event whose status is to be cleared
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-

(table continues...)

Table 100 (continued) Specification for `Dma_ChInterruptDisable` API

Description	This interface may be used by users of DMA channels (e.g. SPI driver) to disable interrupt generation upon occurrence of the specified event.
Source	IFX
Error handling	DMA_E_DRIVER_NOT_INITIALIZED, DMA_E_CHANNEL_NOT_INITIALIZED, DMA_E_CHANNEL_INVALID_EVENT, DMA_E_CHANNEL_INVALID_ID, DMA_E_CORE_CHANNEL_MISMATCH, DMA_E_DATA_TRANSFER_IN_PROGRESS, DMA_E_FREEZE_STATE
Configuration dependencies	-
User hints	-
SFR accessed	DMA_CH_ADICR(w), DMA_TSR(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.19 Dma_GetVersionInfo

Table 101 Specification for `Dma_GetVersionInfo` API

Syntax	`void Dma_GetVersionInfo ( Std_VersionInfoType * const VersionInfoPtr )`	
Service ID	0x12	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	VersionInfoPtr	Pointer where the version information of this Driver is stored
Parameters (in - out)	-	-
Return	void	-
Description	This function provides the version information of the DMA driver	
Source	IFX	
Error handling	DMA_E_NULL_POINTER	
Configuration dependencies	DmaGetVersionInfoApi,ModuleId,VendorId,SwPatchVersion,SwMinorVersion,SwMajorVersion	

(table continues...)

Table 101 (continued) Specification for `Dma_GetVersionInfo` API	
User hints	-
SFR accessed	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.20 Dma_MEStatusClear

Table 102 Specification for `Dma_MEStatusClear` API		

Syntax	`Std_ReturnType  Dma_MEStatusClear`
	`(`
	`  const Dma_MoveEngineListType MoveEngineId`
	`)`	

Service ID	0x13	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	MoveEngineId	The Move Engine number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK if the ME status clear could be done E_NOT_OK: In Case of - Driver is not initialized -Invalid input parameter
Description	This interface may be used by users of DMA channels clear move engine events. The error flags of both the move engines would be cleared by this interface.	
Source	IFX	
Error handling	DMA_E_MOVE_ENGINE_INVALID_ID, DMA_E_DRIVER_NOT_INITIALIZED	
Configuration dependencies	-	
User hints	-	
SFR accessed	CPU_CORE_ID(r), DMA_ME_CLRE(w) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.21 Dma_InitCheck

Table 103 Specification for `Dma_InitCheck` API

Syntax	`Std_ReturnType Dma_InitCheck ( const Dma_ConfigType * const ConfigPtr )`	
Service ID	0x14	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to root data structure of all post build configurations
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Initialization check passed E_NOT_OK: In Case of - Driver is not initialized - ConfigPtr is NULL - ConfigPtr is not same as used for initialization - Prerequisite conditions not satisfied - Global Variables or SFR is not set as expected
Description	This interface is meant to be called by the upper layer using the DMA, to check if the initialization has been successfully performed or not. For the same reason, this interface is expected to be called after the initialization is done. After the checks, the interface would indicate the success or failure in the return value. *Note: This API can be concurrently executed on different cores.*	
Source	IFX	
Error handling	-	
Configuration dependencies	DmaInitCheckApi,DmaDevErrorDetect	
User hints	-	
SFR accessed	CPU_CORE_ID(r), DMA_ACCEN_ACCENR0(r), DMA_CH_ADICR(r), DMA_CH_CHCFGR(r), DMA_CH_CHCSR(r), DMA_CH_DADR(r), DMA_CH_RDCRCR(r), DMA_CH_SADR(r), DMA_CH_SDCRCR(r), DMA_CH_SHADR(r), DMA_CLC(r), DMA_HRR(r), DMA_ME_EER(r), DMA_ME_ERRSR(r), DMA_MODE(r), DMA_PRR0(r), DMA_PRR1(r), DMA_TSR(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.22 Dma_GetCrcValue

Table 104 Specification for `Dma_GetCrcValue` API

Syntax	`uint32 Dma_GetCrcValue`
	`(`
	`  const uint8 ChannelId,`
	`  const Dma_CrcType CrcType`
	`)`	

Service ID	0x15	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	ChannelId CrcType	DMA channel ID Address CRC or Data CRC selection
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	CRC value calculated by the hardware. *Note: The API would return zero if there are any DET/Safety errors. If a zero is returned by the API, the caller should check for the presence of DET/Safety errors.*
Description	This API allows the user to read the CRC value calculated by the DMA hardware.	
Source	IFX	
Error handling	DMA_E_DRIVER_NOT_INITIALIZED, DMA_E_CHANNEL_INVALID_ID, DMA_E_INVALID_CRC_TYPE_REQ, DMA_E_CRC_NOT_SUPPORTED, DMA_E_CORE_CHANNEL_MISMATCH, DMA_E_DATA_TRANSFER_IN_PROGRESS, DMA_E_FREEZE_STATE	
Configuration dependencies	-	
User hints	The 'Dma_GetCrcValue' API provides the CRC value computed by the DMA hardware. Computing the reference CRC and comparing the calculated CRC and reference CRC should be done by the user.	
SFR accessed	DMA_CH_ADICR(r), DMA_CH_RDCRCR(r), DMA_CH_SDCRCR(r), DMA_TSR(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.23 Dma_GetCurrentTimeStamp

Table 105 Specification for `Dma_GetCurrentTimeStamp` API

Syntax	`uint32 Dma_GetCurrentTimeStamp ( void )`	
Service ID	0x16	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	The current timestamp value which can range from 0 to 0xFFFFFFFF. *Note: If a DET or safety error occurs, the timestamp would be returned as zero. If a zero is returned by the API, the caller should check for the presence of any DET or safety errors.*
Description	This API allows the user to read the current time stamp in the DMA hardware. This time stamp can be used to compare against the time stamp appended by the DMA when the data is moved.	
Source	IFX	
Error handling	DMA_E_DRIVER_NOT_INITIALIZED	
Configuration dependencies	-	
User hints	None	
SFR accessed	CPU_CORE_ID(r), DMA_TIME(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.24 Dma_IsChannelInitDone

Table 106 Specification for `Dma_IsChannelInitDone` API

Syntax	`Std_ReturnType Dma_IsChannelInitDone ( const uint8 ChannelId )`	
Service ID	0x17	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	ChannelId	The DMA channel lD
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	The initialization status of the channel E_OK - The initialization is done E_NOT_OK - The initialization is not done *Note: The API would return E_NOT_OK if there are any DET/Safety errors. If E_NOT_OK is returned by the API, the caller should check for the presence of any DET/Safety error.*
Description	This API allows the user to check if the channel being used is in initialized state or not.	
Source	IFX	
Error handling	DMA_E_DRIVER_NOT_INITIALIZED, DMA_E_CHANNEL_INVALID_ID, DMA_E_CORE_CHANNEL_MISMATCH	
Configuration dependencies	-	
User hints	-	
SFR accessed	CPU_CORE_ID(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.25 Dma_SetPattern

Table 107 Specification for `Dma_SetPattern` API

Syntax	`Std_ReturnType Dma_SetPattern ( const uint8 Channel, const uint32 Pattern )`	
Service ID	0x19	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	Channel Pattern	Channel number for which the pattern is intended. This should be the channel which is configured and reserved to be used for pattern detection or conditional linked list feature. The specific pattern which is to be set to the PRR register.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	The status indicating whether the pattern register programming was successful or not. - E_OK - Pattern was programmed into the register - E_NOT_OK - Pattern was not programmed into the register, see Note 3
Description	This API allows the user to set or change the pattern which should be used for the pattern detection/conditional linked list feature. *Note 1: The provided channel should be reserved to use the pattern detection/conditional linked list feature at the precompile time itself. If any other channel number is provided, a DET/Safety Error would be raised.* *Note 2: The provided pattern would be set to the PRR0 or PRR1 register depending on whether the channel is reserved to use the PRR0 register or PRR1 register.* *Note 3: E_NOT_OK will be reported if error is detected. See "Error handling" for the error details.*	
Source	IFX	
Error handling	DMA_E_DRIVER_NOT_INITIALIZED, DMA_E_CHANNEL_INVALID_ID, DMA_E_CHANNEL_NOT_INITIALIZED, DMA_E_CORE_CHANNEL_MISMATCH, DMA_E_DATA_TRANSFER_IN_PROGRESS, DMA_E_CH_PATTERN_INVALID_REQ	
Configuration dependencies	DmaChannelForPattern0,DmaTcsShadowRegisterConfiguration,DmaPatternMode,DmaChannelForPattern1	
User hints	None	

(table continues...)

Table 107 (continued) Specification for `Dma_SetPattern` API

SFR accessed	DMA_PRR0(w), DMA_PRR1(w), DMA_TSR(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.4 Notifications and Callbacks

The DMA driver does not provide any notification or callbacks.

1.3.5 Scheduled functions

The DMA driver does not provide any scheduled functions.

1.3.6 Interrupt service routines

This section lists all the interrupt handlers of the DMA driver.

1.3.6.1 Dma_ChInterruptHandler

Table 108 Specification for `Dma_ChInterruptHandler` API

Syntax	`void Dma_ChInterruptHandler`
	`(`
	`  const uint8 Channel`
	`)`	

Service ID	0xF3	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Conditionally Reentrant - This API is re-entrant for different channels	
Parameters (in)	Channel	DMA channel number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This is the interrupt service routine of a channel interrupt invoked by the interrupt frame (installed in the interrupt vector table). It identifies the cause of the interrupt, clears the status and invokes registered notification routines. *Note: If there are no status flags set in the interrupt registers, the notification function would be given the reason as 'unknown reason'. The user can trigger the appropriate error handling.*	
Source	IFX	

(table continues...)

Table 108 (continued) Specification for `Dma_ChInterruptHandler` API

Error handling	DMA_E_CH_INT_PLAUSIBILITY
Configuration dependencies	-
User hints	-
SFR accessed	CPU_CORE_ID(r), DMA_CH_CHCSR(rw) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.6.2 Dma_MEInterruptDispatcher

Table 109 Specification for `Dma_MEInterruptDispatcher` API

Syntax	`void Dma_MEInterruptDispatcher ( void )`	
Service ID	0xF4	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Not Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This is the interrupt service routine meant to be called by the interrupt frame (installed in the interrupt vector table). The ISR identifies the move engine responsible for the error interrupt and calls the interrupt handler. It also checks the TRL event for channels where 'DmaTcsInterruptTransactionLoss' configuration parameter is enabled.	
Source	IFX	
Error handling	-	
Configuration dependencies	-	
User hints	-	

(table continues...)

Table 109 (continued) Specification for `Dma_MEInterruptDispatcher` API

SFR accessed	CPU_CORE_ID(r), DMA_ME_CLRE(w), DMA_ME_ERRSR(r), DMA_TSR(rw) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.7 Callout

The driver does not support any callout functions.

1.3.8 Errors Handling

This section describes the various errors reported by the DMA driver.

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
Development/Safety Errors: None	None	None	None	None	None
Multicore Errors: None	None	None	None	None	None
DMA_E_NULL_POINTER: The API input parameter is a null pointer	IFX	0x5	DET_SAFETY	0x5	DET_SAFETY
DMA_E_DRIVER_NOT_INITIALIZED: The driver is not initialized using the Dma_Init API or is/being deinitialized using the Dma_Deinit API	IFX	0x1	DET_SAFETY	0x1	DET_SAFETY
DMA_E_CHANNEL_NOT_INITIALIZED: A channel API invoked on a channel before initialization of that channel	IFX	0x2	DET_SAFETY	0x2	DET_SAFETY
DMA_E_CHANNEL_INVALID_ID: Incorrect channel passed as parameter	IFX	0x3	DET_SAFETY	0x3	DET_SAFETY
DMA_E_CHANNEL_INVALID_EVENT: Incorrect channel event	IFX	0x4	DET_SAFETY	0x4	DET_SAFETY
DMA_E_FREEZE_STATE: Data transfer start/channel update requested by application during freeze state	IFX	0x6	DET_SAFETY	0x6	DET_SAFETY
DMA_E_NO_TRANSFERS_PENDING: Freeze requested when no transfers are pending	IFX	0x7	DET_SAFETY	0x7	DET_SAFETY

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
DMA_E_NOT_IN_FREEZE_STATE: Resume requested without a prior freeze	IFX	0x8	DET_SAFETY	0x8	DET_SAFETY
DMA_E_DATA_TRANSFER_IN_PROGRESS: Start transfer requested when a data transfer is already in progress	IFX	0x9	DET_SAFETY	0x9	DET_SAFETY
DMA_E_ALREADY_INITIALIZED: DMA driver initialization requested despite the driver already initialized	IFX	0xA	DET_SAFETY	0xA	DET_SAFETY
DMA_E_CH_ALREADY_INITIALIZED: DMA channel initialization requested despite the channel already initialized	IFX	0xB	DET_SAFETY	0xB	DET_SAFETY
DMA_E_TIMEOUT: Timeout detected while waiting for a certain value of critical register bit fields	IFX	0xC	DET_SAFETY	0xC	DET_SAFETY
DMA_E_CHANNEL_INVALID_SWITCH_REQ: Switch Buffer API invoked despite the feature not configured	IFX	0xD	DET_SAFETY	0xD	DET_SAFETY
DMA_E_INVALID_SHADOW_CONFIG_REQ: Dma_ChUpdate API requested for Shadow register update while ADICR is configured in non-shadow configuration	IFX	0xE	DET_SAFETY	0xE	DET_SAFETY
DMA_E_INVALID_CRC_TYPE_REQ: Dma_GetCrcValue API requested with an invalid CRC type, which is not either Address CRC or Data CRC	IFX	0xF	DET_SAFETY	0xF	DET_SAFETY
DMA_E_CRC_NOT_SUPPORTED: Dma_GetCrcValue API requested for a channel which does not support the CRC calculation	IFX	0x10	DET_SAFETY	0x10	DET_SAFETY
DMA_E_MOVE_ENGINE_INVALID_ID: Incorrect channel passed as parameter	IFX	0x11	DET_SAFETY	0x11	DET_SAFETY

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
DMA_E_CH_NO_NOTIFICATION_CONFIGURED: No notification function is configured for channel interrupt	IFX	0x12	DET_SAFETY	0x12	DET_SAFETY
DMA_E_CHANNEL_INVALID_START_REQ: Dma_ChUpdate API invoked with invalid CHCSR configuration, since SCH bit should not be set.	IFX	0x13	DET_SAFETY	0x13	DET_SAFETY
DMA_E_CH_PATTERN_INVALID_REQ: Pattern configuration invoked for a channel which is not configured for the feature.	IFX	0x14	DET_SAFETY	0x14	DET_SAFETY
DMA_E_CH_INT_PLAUSIBILITY: Error code is reported if an unintended channel interrupt is triggering the ISR.	IFX	0x16	DET_SAFETY	0x16	DET_SAFETY
DMA_E_CORE_NOT_CONFIGURED: Channel not configured for the particular core is invoked	IFX	0x64	DET_SAFETY	0x64	DET_SAFETY
DMA_E_CORE_CHANNEL_MISMATCH: Channel not configured for the particular core is invoked	IFX	0x65	DET_SAFETY	0x65	DET_SAFETY
DMA_E_MASTER_UNINIT: A slave core initialization is attempted, before initializing the master core.	IFX	0x66	DET_SAFETY	0x66	DET_SAFETY
DMA_E_SLAVE_INIT: Error code is reported if de-initialization from master core is invoked while any of the slave cores is still in the initialized state.	IFX	0x67	DET_SAFETY	0x67	DET_SAFETY

1.3.9 Deviations and limitations

The section describes the deviations and limitations of the DMA driver.

1.3.9.1 Deviations

The section describes the deviations of the DMA driver.

1.3.9.1.1 Software specification deviations

The section describes the deviations from software specification.

Table 110 Known deviations

Reference	Deviation
DMA driver writes into the interrupt control registers for the GPSR interrupts.	DMA driver requires accessing the SRC registers of the Interrupt Router in a multicore scenario. The access is needed to route the resource partition interrupts to the appropriate CPU core when an error event occurs.
Handling spurious interrupts	DMA driver does not fulfill the requirement of spurious interrupt handling and will not report any errors for spurious interrupts. DMA driver will gracefully exit if any invalid interrupt occurs. User shall have means to determine if a spurious interrupt has occurred and take appropriate actions. Refer 'Spurious Interrupt handling' AoU.

1.3.9.1.2 AMDC Violations

The DMA driver does not have any AMDC violations.

1.3.9.1.3 VSMD Violations

The DMA driver does not have any VSMD violations.

1.3.9.2 Limitations

The section describes the limitations of the DMA driver.

Table 111 Known limitations

Reference	Limitation
The last TCS node of a linked list, cannot be configured as 'autostart'.	This limitation is applicable only if the linked list feature is being used in the autostart mode. If the linked list feature with autostart is used, the last node must not be configured as autostart. For more information, refer to Chapter 21.3.4.9.3 in the Target Specification.
Global addresses should be used when specifying the addresses to be used by DMA.	When specifying the addresses for the DMA transactions, the addresses to be used should be in the global address range and not in the local address range, as the DMA hardware can handle only the global addresses. User should ensure this when configuring the DMA channels.
Multiple ME interrupts for source and destination errors.	If there are ME errors for source or destination memory, the channel would continue the transaction till all transfers are completed. Because of this, there would be multiple error interrupt notifications for the same transaction.

(table continues...)

Table 111 (continued) Known limitations

Reference	Limitation
Handling of multiple RP error interrupts	As per the hardware design, the DMA IP stores only the last error channel (LEC field of ERRSRm of move engine register) when there are multiple DMA errors. So, if there are multiple DMA RP error interrupts triggered at the same time (faster than the DMA driver RP error ISR processing), there is a possibility that some of the DMA RP error interrupts could be lost. This could result in missing the error notifications to the application which uses DMA. Workaround hint: The application should check for the interrupt overflow flag (IOV bit in SRC register of the DMA RP error interrupt) before invoking the RP error interrupt service routine to the DMA driver. If the IOV flag is set, the application shall take additional measures to evaluate the impact to the users of DMA. (For example, for a missed SPI reception DMA error, the application can monitor the SPI Rx FIFO overflow error flag). The application should clear the IOV bit using the IOVCLR bit of the SRC register, once the handling is complete.
Overloading of DMA channels	When DMA channels are overloaded then there can be a delay of receiving the DMA interrupt, which could further lead to missing of DMA interrupts. In this corner case the driver which is using this DMA channel may lose data or get stuck.

Revision history

Table 112 Revision History

Date	Version	Description
2024-08-09	7.0	Released
2024-08-06	6.1	- Limitation is updated for overloading of DMA channels in section 1.3.9.2.
2023-06-01	6.0	Released
2023-05-23	5.1	- ASIL Level has been updated to Safety level and the description of the safety level is updated in Section 1.3.3 and 1.3.6 - Updated the description of return type E_NOT_OK case information for Dma_InitCheck, Dma_IsInitDone,Dma_MEStatusClear API
2021-11-02	5.0	Released
2021-10-12	4.1	- Error DMA_E_NOT_IN_FREEZE_STATE removed for Dma_ChEnableHardwareTrigger API. - Service ID updated - for Dma_ChInterruptHandler and Dma_MEInterruptDispatcher APIs. - SFR accessed (Updated for Dma_Init, Dma_InitCheck and Dma_DeInit APIs). - Config variant attribute table information is removed and added this information in 'Configuration interfaces' section.
2021-03-25	4.0	Released
2021-03-24	3.1	Updated the description of interrupt connections information Section 1.1.4.5
2020-11-26	3.0	Released
2020-11-26	2.1	Updated the description of Interrupt service routines
2020-11-19	2.0	Released
2020-11-19	1.1	- Updated DMA support for Error handling and Supervision - Updated AoU, Deviations and Interrupt connections for spurious interrupt
2020-08-10	1.0	Released
2020-08-04	0.1	- Initial Version - DMA driver chapter moved from MC-ISAR_TC3xx_UM_CD to this document - ASIL Level updated for Dma_ChInterruptHandler and Dma_MEInterruptDispatcher - DMA Pattern Matching (API: Dma_SetPattern and Configuration Interfaces: DmaPatternConfig, DmaPattern0, DmaPattern1, DmaChannelForPattern0, DmaChannelForPattern1 and DmaPatternMode) and Conditional Linked list (Configuration Interfaces: DmaCLLNextIndex)features added - Dma Deinitialization (API:Dma_DeInit and Configuration Interfaces: DmaDeinitApi) support added - Integration hints added to handle DMA ESM and effiicient handling of DMA interrupts - Configuration parameter DmaChannelUser added to allocate DMA channels to desired driver or user application

Trademarks

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

Edition 2024-08-09

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2024 Infineon Technologies AG

All Rights Reserved.

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-ocr1484806431059

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.

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.