MCAL Configuration Verification Manual for Crc

32-bit TriCore™ AURIX™ TC3xx microcontroller family

About this document

Scope and purpose

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

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

Intended audience

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

Reference documents

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

- AURIX™ TC3xx MCAL User Manual Crc

Table of contents

Table of contents

Crc module

This chapter describes the details of the configuration data generated from the CRC library.

1.1 File: Crc_Cfg.h

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

1.1.1 Macro: CRC_AR_RELEASE_MAJOR_VERSION

Table 1 CRC_AR_RELEASE_MAJOR_VERSION

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

1.1.2 Macro: CRC_AR_RELEASE _MINOR_VERSION

Table 2 CRC_AR_RELEASE _MINOR_VERSION

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

1.1.3 Macro: CRC_AR_RELEASE_REVISION_VERSION

Table 3 CRC_AR_RELEASE_REVISION_VERSION

Name	CRC_AR_RELEASE_REVISION_VERSION

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

1.1.4 Macro: CRC_SW_MAJOR_VERSION

Table 4 **CRC_SW_MAJOR_VERSION**

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

1.1.5 Macro: CRC_SW_MINOR_VERSION

Table 5 **CRC_SW_MINOR_VERSION**

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

1.1.6 Macro: CRC_SW_PATCH_VERSION

Table 6 **CRC_SW_PATCH_VERSION**

Name	CRC_SW_PATCH_VERSION
Description	Patch level version number of the Crc module.

Verification method	The macro is generated with the value present in ‘CommonPublishedInformation/SwPatchVersion’. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Crc_Cfg.h file with SwPatchVersion 0	`#define CRC_SW_PATCH_VERSION (0U)`

1.1.7 Macro: CRC_SAFETYENABLE

Table 7 CRC_SAFETYENABLE

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

1.1.8 Macro: CRC_RUNTIME_API_MODE

Table 8 CRC_RUNTIME_API_MODE

Name	CRC_RUNTIME_API_MODE	
Description	Decides the mode of execution of Run Time API's.	
Verification method	The macro is generated as: - CRC_MCAL_SUPERVISOR if ‘CrcGeneral/CrcRuntimeApiMode’ configuration parameter is set to ‘CRC_MCAL_SUPERVISOR’ - CRC_MCAL_USER1 if ‘CrcGeneral/CrcRuntimeApiMode’ configuration parameter is set to ‘CRC_MCAL_USER1’ *Note: CrcRuntimeApiMode will be available only when at least one of the DMA based APIs is enabled, otherwise the parameter is editable false.*	
Example(s)	**Action**	**Generated output**
	CrcRuntimeApiMode = CRC_MCAL_SUPERVISOR	`#define CRC_RUNTIME_API_MODE (CRC_MCAL_SUPERVISOR)`
	CrcRuntimeApiMode = CRC_MCAL_USER1	`#define CRC_RUNTIME_API_MODE (CRC_MCAL_USER1)`

1.1.9 Macro: CRC_VERSION_INFO_API

Table 9 CRC_VERSION_INFO_API

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

1.1.10 Macro: CRC_16ARC_MODE

Table 10 CRC_16ARC_MODE

Name	CRC_16ARC_MODE	
Description	Switch to select one of the available CRC 16-bit (0x8005h) calculation methods.	
Verification method	The macro is generated as: - CRC_HARDWARE_MODE if 'Crc16ARCMode' configuration container is set to CRC_16_ARC_HARDWARE - Else generated as CRC_RUNTIME_MODE if 'Crc16ARCMode' configuration container is set to CRC_16_ARC_RUNTIME - Else generated as CRC_TABLE_MODE if 'Crc16ARCMode' configuration container is set to CRC_16_TABLE - Else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	Container Crc16ARCMode= CRC_16_ARC_HARDWARE	`#define CRC_16ARC_MODE (CRC_HARDWARE_MODE)`
	Container Crc16ARCMode= CRC_16_ARC_RUNTIME	`#define CRC_16ARC_MODE (CRC_RUNTIME_MODE)`
	Container Crc16ARCMode= CRC_16_ARC_TABLE	`#define CRC_16ARC_MODE (CRC_TABLE_MODE)`
	Container Crc16ARCMode= empty	`#define CRC_16ARC_MODE (STD_OFF)`

1.1.11 Macro: CRC_16_MODE

Table 11 CRC_16_MODE

Name	CRC_16_MODE	
Description	Selection of 16 bit CRC method (based on CCITT-FALSE CRC16 0x1021 Polynomial).	
Verification method	The macro is generated as : - CRC_HARDWARE_MODE if 'Crc16Mode' configuration container is set to CRC_16_HARDWARE - Else generated as CRC_RUNTIME_MODE if 'Crc16Mode' configuration container is set to CRC_16_RUNTIME - Else generated as CRC_TABLE_MODE if 'Crc16Mode' configuration container is set to CRC_16_TABLE - Else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	Container Crc16Mode = CRC_16_HARDWARE	`#define CRC_16_MODE (CRC_HARDWARE_MODE)`
	Container Crc16Mode = CRC_16_RUNTIME	`#define CRC_16_MODE (CRC_RUNTIME_MODE)`
	Container Crc16Mode = CRC_16_TABLE	`#define CRC_16_MODE (CRC_TABLE_MODE)`
	Container Crc16Mode = empty	`#define CRC_16_MODE (STD_OFF)`

1.1.12 Macro: CRC_32_MODE

Table 12 CRC_32_MODE

Name	CRC_32_MODE	
Description	Selection of 32 bit CRC method (based on IEEE-802.3 CRC32 0x04C11DB7 Polynomial).	
Verification method	The macro is generated as: - CRC_HARDWARE_MODE if 'Crc32Mode' configuration container is set to CRC_32_HARDWARE - Else generated as CRC_RUNTIME_MODE if 'Crc32Mode' configuration container is set to CRC_32_RUNTIME - Else generated as CRC_TABLE_MODE if 'Crc32Mode' configuration container is set to CRC_32_TABLE - Else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	Container Crc32Mode = CRC_32_RUNTIME	`#define CRC_32_MODE (CRC_RUNTIME_MODE)`
	Container Crc32Mode = CRC_32_TABLE	`#define CRC_32_MODE (CRC_TABLE_MODE)`
	Container Crc32Mode = empty	`#define CRC_32_MODE (STD_OFF)`

1.1.13 Macro: CRC_32P4_MODE

Table 13 CRC_32P4_MODE

Name	CRC_32P4_MODE	
Description	Selection of 32 bit CRCP4 method (based on 0xF4ACFB13 Polynomial).	
Verification method	The macro is generated as : - CRC_RUNTIME_MODE if 'Crc32P4Mode' configuration container is set to CRC_32P4_RUNTIME - Else generated as CRC_TABLE_MODE if 'Crc32P4Mode' configuration container is set to CRC_32P4_TABLE - Else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	Container Crc32P4Mode = CRC_32P4_RUNTIME	`#define CRC_32P4_MODE (CRC_RUNTIME_MODE)`
	Container Crc32P4Mode = CRC_32P4_TABLE	`#define CRC_32P4_MODE (CRC_TABLE_MODE)`
	Container Crc32P4Mode = empty	`#define CRC_32P4_MODE (STD_OFF)`

1.1.14 Macro: CRC_8_MODE

Table14 CRC_8_MODE

Name	CRC_8_MODE	
Description	Selection of 8 bit CRC method (based on SAE-J1850 CRC8 0x1D Polynomial).	
Verification method	The macro is generated as: - CRC_HARDWARE_MODE if 'Crc8Mode' configuration container is set to CRC_8_HARDWARE - Else generated as CRC_RUNTIME_MODE if 'Crc8Mode' configuration container is set to CRC_8_RUNTIME - Else generated as CRC_TABLE_MODE if 'Crc8Mode' configuration container is set to CRC_8_TABLE - Else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	Container Crc8Mode = CRC_8_HARDWARE	`#define CRC_8_MODE (CRC_HARDWARE_MODE)`
	Container Crc8Mode = CRC_8_RUNTIME	`#define CRC_8_MODE (CRC_RUNTIME_MODE)`
	Container Crc8Mode = CRC_8_TABLE	`#define CRC_8_MODE (CRC_TABLE_MODE)`

	Container Crc8Mode = empty	`#define CRC_8_MODE (STD_OFF)`

1.1.15 Macro: CRC_8H2F_MODE

Table 15 CRC_8H2F_MODE

Name	CRC_8H2F_MODE	
Description	Selection of 8 bit CRC method (based on 0x2F Polynomial).	
Verification method	The macro is generated as: - CRC_HARDWARE_MODE if 'Crc8H2FMode' configuration container is set to CRC_8H2F_HARDWARE - Else generated as CRC_RUNTIME_MODE if 'Crc8H2FMode' configuration container is set to CRC_8H2F_RUNTIME - Else generated as CRC_TABLE_MODE if 'Crc8H2FMode' configuration container is set to CRC_8H2F_TABLE - Else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	Container Crc8H2FMode = CRC_8H2F_HARDWARE	`#define CRC_8H2F_MODE (CRC_HARDWARE_MODE)`
	Container Crc8H2FMode = CRC_8H2F_RUNTIME	`#define CRC_8H2F_MODE (CRC_RUNTIME_MODE)`
	Container Crc8H2FMode = CRC_8H2F_TABLE	`#define CRC_8H2F_MODE (CRC_TABLE_MODE)`
	Container Crc8H2FMode = empty	`#define CRC_8H2F_MODE (STD_OFF)`

1.1.16 Macro: CRC_64_MODE

Table 16 CRC_64_MODE

Name	CRC_64_MODE	
Description	Switch to select one of the available CRC 64-bit calculation methods. *Note: CRC64 does not support CRC calculation in the hardware mode (CRC_64_HARDWARE) as the current FCE engine does not provide the kernel for CRC64.*	
Verification method	The macro is generated as: - CRC_64_RUNTIME if 'Crc64Mode' configuration container is set to CRC_64_RUNTIME - Else generated as CRC_TABLE_MODE if 'Crc64Mode' configuration container is set to CRC_64_TABLE - Else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**

	Container Crc64Mode = CRC_64_RUNTIME	`#define CRC_64_MODE (CRC_RUNTIME_MODE)`
	Container Crc64Mode = CRC_64_TABLE	`#define CRC_64_MODE (CRC_TABLE_MODE)`
	Container Crc64Mode = empty	`#define CRC_64_MODE (STD_OFF)`

1.1.17 Macro: CRC_8_DMAMODE

Table 17 CRC_8_DMAMODE

Name	CRC_8_DMAMODE	
Description	Pre-processor switch to enable / disable CrcDma8bitApi for the polynomial CRC8. True: Crc_DmaCalculateCRC8 API enable. False: Crc_DmaCalculateCRC8 API disable. The optional APIs are disabled by default to minimize the executable code size.	
Verification method	The macro is generated as: - STD_ON if 'CrcDma8bitApi' configuration is enabled - Else generated as STD_OFF if 'CrcDma8bitApi' configurationis disabled.	
Example(s)	**Action**	**Generated output**
	Container CrcDma8bitApi= True	`#define CRC_8_DMAMODE (STD_ON)`
	Container CrcDma8bitApi = False	`#define CRC_8_DMAMODE (STD_OFF)`

1.1.18 Macro: CRC_16_DMAMODE

Table 18 CRC_16_DMAMODE

Name	CRC_16_DMAMODE	
Description	Pre-processor switch to enable / disable CrcDma16bitApi for the polynomial CRC16. True: Crc_DmaCalculateCRC16 API enable. False: Crc_DmaCalculateCRC16 API disable. The optional APIs are disabled by default to minimize the executable code size.	
Verification method	The macro is generated as: - STD_ON if 'CrcDma16bitApi' configuration is enabled - Else generated as STD_OFF if 'CrcDma16bitApi' configurationis disabled.	
Example(s)	**Action**	**Generated output**
	Container CrcDma16bitApi= True	`#define CRC_16_DMAMODE (STD_ON)`
	Container CrcDma16bitApi = False	`#define CRC_16_DMAMODE (STD_OFF)`

1.1.19 Macro: CRC_32_DMAMODE

Table 19 CRC_32_DMAMODE

Name	CRC_32_DMAMODE	
Description	Pre-processor switch to enable / disable CrcDma32bitApi for the polynomial CRC32. True: Crc_DmaCalculateCRC32 API enable. False: Crc_DmaCalculateCRC32 API disable. The optional APIs are disabled by default to minimize the executable code size.	
Verification method	The macro is generated as: - STD_ON if 'CrcDma32bitApi' configuration is enabled - Else generated as STD_OFF if 'CrcDma32bitApi' configurationis disabled.	
Example(s)	**Action**	**Generated output**
	Container CrcDma32bitApi= True	`#define CRC_32_DMAMODE (STD_ON)`
	Container CrcDma32bitApi = False	`#define CRC_32_DMAMODE (STD_OFF)`

1.1.20 Macro: CRC_32P4_DMAMODE

Table 20 CRC_32P4_DMAMODE

Name	CRC_32P4_DMAMODE	
Description	Pre-processor switch to enable / disable CrcDma32P4bitApi for the polynomial CRC16. True: Crc_DmaCalculateCRC32P4 API enable. False: Crc_DmaCalculateCRC32P4 API disable. The optional APIs are disabled by default to minimize the executable code size.	
Verification method	The macro is generated as: - STD_ON if 'CrcDma32P4bitApi' configuration is enabled - Else generated as STD_OFF if 'CrcDma32P4bitApi' configurationis disabled.	
Example(s)	**Action**	**Generated output**
	Container CrcDma32P4bitApi= True	`#define CRC_32P4_DMAMODE (STD_ON)`
	Container CrcDma32P4bitApi = False	`#define CRC_32P4_DMAMODE (STD_OFF)`

1.1.21 Macro: CRC_DMA_MAX_CHANNELS

Table 21 CRC_DMA_MAX_CHANNELS

Name	CRC_DMA_MAX_CHANNELS	
Description	The maximum number of the DMA channels configured in the CRC module	
Verification method	The macro is generated as a numeric value which corresponds to the total number of DMA channels configured. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Three cores are allocated with the DMA reosurces.	`#define CRC_DMA_MAX_CHANNELS (3U)`

1.1.22 Macro: CRC_CONFIGURED_CORE[x]

Table 22 CRC_CONFIGURED_CORE[x]

Name	CRC_CONFIGURED_CORE[x]	
Description	This parameter provides the configured COREs. x represent the core number.	
Verification method	The macro is generated as: - STD_ON if the core[x] is assigned with the FCE and DMA resources. Else generated as STD_OFF if the core[x] is not assigned with any resources.	
Example(s)	**Action**	**Generated output**
	Core0 is assigned with the resources	`#define CRC_CONFIGURED_CORE0 (STD_ON)`
	Core3 is not assigned with any resources	`#define CRC_CONFIGURED_CORE3 (STD_OFF)`

1.1.23 Macro: CRC_16ARC_CONFIGERROR_VAL

Table 23 CRC_16ARC_CONFIGERROR_VAL

Name	CRC_16ARC_CONFIGERROR_VAL	
Description	16 bit error return value for Crc_CalculateCRC16ARC API.	
Verification method	The macro is generated as a numeric value set in the configuration parameter 'CrcGeneral/Crc16ARCReturnErrorValue'.	
Example(s)	**Action**	**Generated output**
	Set Crc16ARCReturnErrorValue as 0	`#define CRC_16_CONFIGERROR_VAL (0U)`

	Set Crc16ARCReturnErrorValue as 240	`#define CRC_16_CONFIGERROR_VAL (240U)`

1.1.24 Macro: CRC_8_CONFIGERROR_VAL

Table 24 CRC_8_CONFIGERROR_VAL

Name	CRC_8_CONFIGERROR_VAL	
Description	8 bit error return value for Crc_CalculateCRC8 API.	
Verification method	The macro is generated as a numeric value set in the configuration parameter 'CrcGeneral/Crc8ReturnErrorValue'.	
Example(s)	**Action**	**Generated output**
	Set Crc8ReturnErrorValue as 0	`#define CRC_8_CONFIGERROR_VAL (0U)`
	Set Crc8ReturnErrorValue as 240	`#define CRC_8_CONFIGERROR_VAL (240U)`

1.1.25 Macro: CRC_8H2F_CONFIGERROR_VAL

Table 25 CRC_8H2F_CONFIGERROR_VAL

Name	CRC_8H2F_CONFIGERROR_VAL	
Description	8 bit error return value for Crc_CalculateCRC8H2F API.	
Verification method	The macro is generated as a numeric value set in the configuration parameter 'CrcGeneral/Crc8H2FReturnErrorValue'.	
Example(s)	**Action**	**Generated output**
	Set Crc8H2FReturnErrorValue as 0	`#define CRC_8H2F_CONFIGERROR_VAL (0U)`
	Set Crc8H2FReturnErrorValue as 150	`#define CRC_8H2F_CONFIGERROR_VAL (150U)`

1.1.26 Macro: CRC_16_CONFIGERROR_VAL

Table 26 CRC_16_CONFIGERROR_VAL

Name	CRC_16_CONFIGERROR_VAL	
Description	16 bit error return value for Crc_CalculateCRC16 API.	
Verification method	The macro is generated as a numeric value set in the configuration parameter 'CrcGeneral/Crc16ReturnErrorValue'.	
Example(s)	**Action**	**Generated output**
	Set Crc16ReturnErrorValue as 0	`#define CRC_16_CONFIGERROR_VAL (0U)`

	Set Crc16ReturnErrorValue as 200	`#define CRC_16_CONFIGERROR_VAL (200U)`

1.1.27 Macro: CRC_32_CONFIGERROR_VAL

Table 27 CRC_32_CONFIGERROR_VAL

Name	CRC_32_CONFIGERROR_VAL	
Description	32 bit error return value for Crc_CalculateCRC32 API.	
Verification method	The macro is generated as a numeric value set in the configuration parameter 'CrcGeneral/Crc32ReturnErrorValue'.	
Example(s)	**Action**	**Generated output**
	Set Crc32ReturnErrorValue as 0	`#define CRC_32_CONFIGERROR_VAL (0U)`
	Set Crc32ReturnErrorValue as 150	`#define CRC_32_CONFIGERROR_VAL (150U)`

1.1.28 Macro: CRC_32P4_CONFIGERROR_VAL

Table 28 CRC_32P4_CONFIGERROR_VAL

Name	CRC_32P4_CONFIGERROR_VAL	
Description	32 bit error return value for Crc_CalculateCRC32P4 API.	
Verification method	The macro is generated as a numeric value set in the configuration parameter 'CrcGeneral/Crc32P4ReturnErrorValue'.	
Example(s)	**Action**	**Generated output**
	Set Crc32P4ReturnErrorValue as 0	`#define CRC_32P4_CONFIGERROR_VAL (0U)`
	Set Crc32P4ReturnErrorValue as 200	`#define CRC_32P4_CONFIGERROR_VAL (200U)`

1.1.29 Macro: CRC_64_CONFIGERROR_VAL

Table 29 CRC_64_CONFIGERROR_VAL

Name	CRC_64_CONFIGERROR_VAL	
Description	64 bit error return value for Crc_CalculateCRC64 API.	
Verification method	The macro is generated as a numeric value set in the configuration parameter 'CrcGeneral/Crc64ReturnErrorValue'.	
Example(s)	**Action**	**Generated output**
	Set Crc64ReturnErrorValue as 0	`#define CRC_64_CONFIGERROR_VAL (0U)`
	Set Crc64ReturnErrorValue as 200	`#define CRC_64_CONFIGERROR_VAL (200U)`

1.2 File: Crc_Cfg.c

1.2.1 Structure: Crc_ChannelConfig_Core<x>

Table 30 Crc_ChannelConfig_Core<x>

Name	Crc_ChannelConfig_Core<x>	
Type	Crc_ChannelConfigType	
Description	Configuration structure for the resource configured in the CRC module for each core.	
Verification method	The generated structure will be present in Crc_Cfg.c file. The contents of each member of the structure is explained in the further sections. *Note:* Crc_ChannelConfig_Core<x> *will be available only when at least one of the DMA based API is enabled.*	
Example(s)	**Action**	**Generated output**

	• Configure only one resource in core0 with following resource configuration settings
	1. *Fce_Channel*
	2. *Dma_Channel*
	3. *ResNotificationPtr*
	4. *ErrNotificationPtr*
	
	Output is shown for Core 0
		`static const Crc_ChannelConfigType Crc_ChannelConfig_Core0=`
		`{`
		`  /*FCE channel number*/`
		`  1U,`
		`  /*DMA channel number*/`
		`  2U,`
		`  /*Result handler configured by the user for the channel*/`
		`  App_CrcNofitCore0,`
		`  /*Error handler configured by the user for the channel*/`

		`App_CrcNofitErrCore0`
		`};`

1.2.1.1 Member: Fce_Channel

Table 31 Fce_Channel

Name	Fce_Channel	
Type	uint8	
Description	This element contains the FCE hardware channel number for the core.	
Verification method	Configure the FCE channel number in /Crc/CrcChannelConfig/CrcChannelId	
Example(s)	**Action**	**Generated output**

	Configure CrcChannelId = 0	`/*FCE channel number*/`
		`0U,`

	Configure CrcChannelId = 2	`/*FCE channel number*/`
		`2U,`

1.2.1.2 Member: Dma_Channel

Table 32 Dma_Channel

Name	Dma_Channel	
Type	uint8	
Description	This element contains the reference to DMA resource allocated to FCE channel for each core.	
Verification method	Select the Dma channel from the drop-down list /Crc/CrcChannelConfig/CrcDmaChannel	
Example(s)	**Action**	**Generated output**

	Configure CrcDmaChannel = /Dma/Dma/DmaChannelConfig_0	`/*DMA channel number*/`
		`0U,`

	Configure CrcDmaChannel = /Dma/Dma/DmaChannelConfig_3	`/*DMA channel number*/`
		`3U,`

1.2.1.3 Member: ResNotificationPtr

Table 33 ResNotificationPtr

Name	ResNotificationPtr	
Type	Crc_ResNotificationPtrType	
Description	This is a user defined result notification callback function will be invoked by the CRC driver on successful CRC calculation.	
Verification method	The generated notification function will be present in Crc_Cfg.c file. Configure the user callback function in /Crc/CrcChannelConfig/CrcResultNotification.	
Example(s)	**Action**	**Generated output**
	Configure CrcResultNotification = App_CrcNofitCore0	`/*Result handler configured by the user for the channel*/` `App_CrcNofitCore0,`
	Configure CrcResultNotification = App_CrcNofitCore3	`/*Result handler configured by the user for the channel*/` `App_CrcNofitCore3,`

1.2.1.4 Member: ErrNotificationPtr

Table 34 ErrNotificationPtr

Name	ErrNotificationPtr	
Type	Crc_ErrNotificationPtrType	
Description	This is the user defined error notification callback function will be invoked by the CRC driver, when an error occured due to move engine.	
Verification method	The generated notification function will be present in Crc_Cfg.c file. Configure the user callback function in /Crc/CrcChannelConfig/CrcErrorNotification.	
Example(s)	**Action**	**Generated output**
	Configure CrcErrorNotification = App_CrcNofitErrCore0	`/*Error handler configured by the user for the channel*/` `App_CrcNofitErrCore0,`
	Configure CrcErrorNotification = App_CrcNofitErrCore3	`/*Error handler configured by the user for the channel*/` `App_CrcNofitErrCore3,`

1.2.2 Array: Crc_ChannelConfig[x]

Table 35 Crc_ChannelConfig[x]

Name	Crc_ChannelConfig[X]

Type	Crc_ChannelConfigType	
Description	Array holds the address of CRC channel configuration specific to each core.	
Verification method	The generated array will be present in Crc_Cfg.c file. Configure the Crc channel in /Crc/CrcChannelConfig If the Crc channel is configured, The Crc_ChannelConfig_Core<x> address will be referenced for each core otherwise NULL_PTR will be present. *Note: Crc_ChannelConfig will be available only when at least one of the DMA based API is enabled.*	
Example(s)	**Action**	**Generated output**

	Only core0 and core1 configured for variant TC399x	`const Crc_ChannelConfigType *const Crc_ChannelConfig[6] =`
		`{`
		`  /*Config parameter for Core0*/`
		`  &Crc_ChannelConfig_Core0,`
		`  /*Config parameter for Core1*/`
		`  &Crc_ChannelConfig_Core1,`
		`  /*Core2 is not configured or not available*/`
		`  NULL_PTR,`
		`  /*Core3 is not configured or not available*/`
		`  NULL_PTR,`
		`  /*Core4 is not configured or not available*/`
		`  NULL_PTR,`
		`  /*Core5 is not configured or not available*/`
		`  NULL_PTR,`
		`};`

1.2.3 Array: Crc_Table32P4[CRC_TABLE_LENGTH]

Table 36 Crc_Table32P4[CRC_TABLE_LENGTH]

Name	Crc_Table32P4[CRC_TABLE_LENGTH]
Type	uint32
Description	The array contains the lookup table for CRC32P4 calculation. (table method)
Verification method	The generated array will be present in Crc_Cfg.c file. Select the calculation mode from the drop-down list /Crc/CrcGeneral/Crc32P4Mode

	Note: The table will be available only when the mode is selected to table method.	
Example(s)	**Action**	**Generated output**

	Configure Crc32P4Mode = CRC_32P4_TABLE	`const uint32 Crc_Table32P4[256] =`
		`{`
		`0x00000000U, 0xF4ACFB13U, 0x1DF50D35U, 0xE959F626U,`
		`0x3BEA1A6AU, 0xCF46E179U, 0x261F175FU, 0xD2B3EC4CU,`
		`0x77D434D4U, 0x8378CFC7U, 0x6A2139E1U, 0x9E8DC2F2U,`
		`...`
		`...`
		`};`

1.2.4 Array: Crc_Table64 [CRC_TABLE_LENGTH]

Table 37 Crc_Table64 [CRC_TABLE_LENGTH]

Name	Crc_Table64[CRC_TABLE_LENGTH]	
Type	uint64	
Description	The array contains the lookup table for CRC64 calculation. (table method)	
Verification method	The generated array will be present in Crc_Cfg.c file. Select the calculation mode from the drop-down list /Crc/CrcGeneral/Crc64Mode *Note: The table will be available only when the mode is selected to table method.*	
Example(s)	**Action**	**Generated output**

	Configure Crc64Mode = CRC_64_TABLE	`const uint64 Crc_Table64[256] =`
		`{`
		`0x0000000000000000U, 0x42F0E1EBA9EA3693U, 0x85E1C3D753D46D26U, 0xC711223CFA3E5BB5U,`
		
		`0x493366450E42ECDFU, 0x0BC387AEA7A8DA4CU, 0xCCD2A5925D9681F9U, 0x8E224479F47CB76AU,`
		`...`
		`...`
		`};`

Revision history

Major changes since the last revision

Date	Version	Description
2020-12-07	V3.0	Document released
2020-12-04	V2.1	For inspection • Crc driver chapter moved from MC-ISAR_TC3xx_Config_Verification_Manual_Basic.pdf to this document. • Macros and structures are added for AS440 changes and DMA based APIs.
2019-07-19	V2.0	No change identified from previous releases. Only version and date change
2019-02-18	V1.10.0_1.0	Initial Release

Edition 2020-12-07
Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
Doc_Number