User Manual

for S32K14X CRCU Driver

Document Number: UM2CRCUASR4.2 Rev0002R1.0.2
Rev. 1.0

Contents

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

Chapter 1
Revision History

Chapter 2
Introduction

2.1 Supported Derivatives....9

2.2 Overview....9

2.3 About this Manual....10

2.4 Acronyms and Definitions....10

2.5 Reference List....11

Chapter 3
Driver

3.1 Requirements....13

3.2 Driver Design Summary....13

3.3 Hardware Resources....13

3.4 Deviation from Requirements....14

3.5 CRCU Driver Limitations....14

3.6 Driver Usage and Configuration Tips....14

3.6.1 CRC Unit Channel Concept....14

3.6.2 CRC Unit Initialization....14

3.6.3 Computing CRC using CRC Unit....15

3.6.4 CRC Unit Channel Configuration....15

3.6.5 CRC Unit Channel Writing Initial Seed Value....15

3.6.6 CRC Unit Synchronous Channel CRC Calculation....16

3.6.7 CRC Unit Asynchronous Channel CRC Calculation....16

3.6.8 CRC Unit Channel Reading CRC Signature....16

3.6.9 CRCUExamples for using the driver....17

3.6.9.1 CCITT-FALSE CRC16 Example....17

3.6.9.2 CRC-32 IEEE 802.3 Example....17

3.7 Runtime Errors....................18

3.8 Software specification....................19

3.8.1 Define Reference....................19

3.8.1.1 Define CRCU_INSTANCE_ID....................19

3.8.1.2 Define CRCU_INIT_ID....................20

3.8.1.3 Define CRCU_SETCHANNELCONFIG_ID....................20

3.8.1.4 Define CRCU_SETCHANNELSEED_ID....................20

3.8.1.5 Define CRCU_SYNCCALCULATECHANNELCRC_ID....................21

3.8.1.6 Define CRCU_ASYNCCALCULATECHANNELCRC_ID....................21

3.8.1.7 Define CRCU_GETCHANNELCRC_ID....................21

3.8.1.8 Define CRCU_GETVERSIONINFO_ID....................22

3.8.1.9 Define CRCU_E_UNINIT....................22

3.8.1.10 Define CRCU_E_INVALID_CHANNEL....................23

3.8.1.11 Define CRCU_E_INVALID_CHANNEL_CONFIG....................23

3.8.1.12 Define CRCU_E_INVALID_POINTER....................23

3.8.1.13 Define CRCU_E_INVALID_LENGTH....................24

3.8.1.14 Define CRCU_E_DMA_CH_NOT_CONFIGURED....................24

3.8.1.15 Define CRCU_E_CHANNEL_BUSY....................24

3.8.1.16 Define CRCU_DEV_ERROR_DETECT....................25

3.8.1.17 Define CRCU_PRECOMPILE_SUPPORT....................25

3.8.1.18 Define CRCU_GET_VERSION_INFO_API....................25

3.8.1.19 Define CRCU_DMA_USED....................26

3.8.1.20 Define CrcuConf_CrcuChannel_CrcuChannel_0....................26

3.8.1.21 Define CrcuConf_CrcuChannelConfig_CrcuChannelConfig_0....................26

3.8.2 Enum Reference....................27

3.8.2.1 Structure Crcu_ChannelStateType....................27

3.8.3 Function Reference....................27

3.8.3.1 Function Crcu_Init....................27

3.8.3.2 Function Crcu_SetChannelConfig....................28

3.8.3.3 Function Crcu_SetChannelSeed........ 28
3.8.3.4 Function Crcu_SyncCalculateChannelCrc........ 29
3.8.3.5 Function Crcu_AsyncCalculateChannelCrc........ 29
3.8.3.6 Function Crcu_GetChannelCrc........ 30
3.8.3.7 Function Crcu_GetVersionInfo........ 30
3.8.4 Structs Reference........ 31
3.8.4.1 Structure Crcu_ChannelType........ 31
3.8.4.2 Structure Crcu_ConfigType........ 31
3.8.5 Types Reference........ 32
3.8.5.1 Typedef Crcu_ChannelIdType........ 32
3.8.5.2 Typedef Crcu_ChannelConfigIdType........ 33
3.8.5.3 Typedef Crcu_ChannelAddressType........ 33
3.8.5.4 Typedef Crcu_ValueType........ 33
3.8.5.5 Typedef Crcu_AsyncCalculateCrcNotifType........ 34
3.9 Symbolic Names DISCLAIMER........ 34

Chapter 4
Tresos Configuration Plug-in

4.1 Configuration elements of Crcu........ 35
4.2 Form IMPLEMENTATION_CONFIG_VARIANT........ 35
4.3 Form CrcuGeneral........ 36
4.3.1 CrcuDevErrorDetect (CrcuGeneral)........ 36
4.3.2 CrcuVersionInfoApi (CrcuGeneral)........ 36
4.3.3 CrcuDmaUsed (CrcuGeneral)........ 37
4.3.4 CrcuEnableUserModeSupport (CrcuGeneral)........ 37
4.4 Form CrcuConfig........ 38
4.4.1 Form CrcuChannel........ 38
4.4.1.1 CrcuChannelId (CrcuChannel)........ 39
4.4.1.2 CrcChannel (CrcuChannel)........ 39
4.4.1.3 CrcuAsyncCalculateCrcCompleteNotification (CrcuChannel)........ 40

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

4.4.1.4 CrcuDmaChannelRef (CrcuChannel).......... 40

4.4.2 Form CrcuChannelConfig.......... 40

4.4.2.1 CrcuChannelConfigId (CrcuChannelConfig).......... 41

4.4.2.2 Crcu_Width (CrcuChannelConfig).......... 41

4.4.2.3 Crcu_Polynom (CrcuChannelConfig).......... 42

4.4.2.4 Crcu_In_Swap (CrcuChannelConfig).......... 42

4.4.2.5 Crcu_Out_Swap (CrcuChannelConfig).......... 43

4.4.2.6 Crcu_Out_Inversion (CrcuChannelConfig).......... 43

4.5 Form CommonPublishedInformation.......... 43

4.5.1 ArReleaseMajorVersion (CommonPublishedInformation).......... 44

4.5.2 ArReleaseMinorVersion (CommonPublishedInformation).......... 44

4.5.3 ArReleaseRevisionVersion (CommonPublishedInformation).......... 45

4.5.4 ModuleId (CommonPublishedInformation).......... 45

4.5.5 SwMajorVersion (CommonPublishedInformation).......... 46

4.5.6 SwMinorVersion (CommonPublishedInformation).......... 46

4.5.7 SwPatchVersion (CommonPublishedInformation).......... 47

4.5.8 VendorApiInfix (CommonPublishedInformation).......... 47

4.5.9 VendorId (CommonPublishedInformation).......... 47

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0	26/04/2019	NXP MCAL Team	Updated version for ASR 4.2.2S32K14XR1.0.2

Chapter 2
Introduction

This User Manual describes NXP Semiconductors AUTOSAR Cyclic Redundancy Check Unit (CRCU) for S32K14X .

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

2.1 Supported Derivatives

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

Table 2-1. S32K14X Derivatives

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

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

2.2 Overview

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

AUTOSAR

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

2.3 About this Manual

This Technical Reference employs the following typographical conventions:

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

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

Notes and warnings are shown as below:

Note

This is a note.

2.4 Acronyms and Definitions

Table 2-2. Acronyms and Definitions

Term	Definition
API	Application Programming Interface
ASM	Assembler
AUTOSAR	Automotive Open System Architecture
CDD	Complex Device Driver
CRC	Cyclic Redundancy Check
CRCU	CRC Unit
DEM	Diagnostic Event Manager
DET	Development Error Tracer
N/A	Not Applicable
MCU	Micro Controller Unit

2.5 Reference List

Table 2-3. Reference List

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

Chapter 3
Driver

3.1 Requirements

List of requirements for CRCU driver can be found in the TraceabilityMatrix document.

3.2 Driver Design Summary

The CRCU driver is implemented as an complex device driver. It uses the CRC hardware peripheral which provides support for implementing the CRC calculations.

The driver offers a hardware independent API to the upper layer that can be used to configure the CRCU and initiate CRC calculations.

Hardware and software settings can be configured using an Autosar standard configuration tool. The information required for a CRC data calculation will be configured in a data structure that will be sent as parameter to the API of the driver.

3.3 Hardware Resources

Table 3-1. CRCU Hardware availability for S32K14X family

Device	Total CRC units
s32k118_lqfp48	1
s32k118_lqfp64	1
s32k142_lqfp64	1
s32k142_lqfp100	1
s32k144_lqfp64	1
s32k144_lqfp100	1
s32k144_mapbga100	1

Table continues on the next page...

Table 3-1. CRCU Hardware availability for S32K14X family (continued)

s32k146_lqfp64	1
s32k146_lqfp100	1
s32k146_mapbga100	1
s32k146_lqfp144	1
s32k148_mapbga100	1
s32k148_lqfp144	1
s32k148_lqfp176	1

3.4 Deviation from Requirements

N/A

3.5 CRCU Driver Limitations

None.

3.6 Driver Usage and Configuration Tips

This driver is an Complex Device Driver. Complete driver functionality together with API description can be found below.

3.6.1 CRC Unit Channel Concept

The expression **channel** in the following text means the identifier of the CRC **unit**. The S32K14X device family has one CRC hardware unit(s).

The channel number is computed in the following way.

- CRC Unit 0 => Channel 0

3.6.2 CRC Unit Initialization

The `Crcu_Init()` function shall initialize the CRCU hardware peripheral(s) and the internal driver context, according to the input configuration data. The application shall ensure that the `Crcu_Init()` function is called first. Only the `Crcu_GetVersionInfo()` can be called before `Crcu_Init()`.

3.6.3 Computing CRC using CRC Unit

After initializing the CRCU driver, there are 2 different methods to compute a CRC:

a) Provide data to the driver via a pointer and a length:

1. Configure the CRCU channel by calling Crcu_SetChannelConfig().
2. Set the CRCU channel seed by calling Crcu_SetChannelSeed().
3. Call function CalculateChannelCrc() and use the returned value as the result of the CRC computation.

In order to apply the same CRC algorithm on a different data string, repeat steps 2-3.

b) Transfer data directly to the CRC Data register:

1. Configure the CRCU channel by calling Crcu_SetChannelConfig().
2. Set the CRCU channel seed by calling Crcu_SetChannelSeed().
3. Transfer data to adress returned by Crcu_SetChannelConfig().
4. Call function Crcu_GetChannelCrc() to get the result of the CRC algorithm.

In order to apply the same CRC algorithm on a different data string, repeat steps 2-4.

3.6.4 CRC Unit Channel Configuration

Driver provides functionality for CRCU channel configuration. The `Crcu_SetChannelConfig()` function is responsible for channel configuration, according to input parameters. This function configures channel's CRC width, polynomial and whether Swap and/or Inversion functionality is applied on CRC input data and/or signature or not.

The `Crcu_SetChannelConfig()` funtion returns pointer to the CRC channel register where external application is able to write data for CRC checksum computation directly.

3.6.5 CRC Unit Channel Writing Initial Seed Value

Driver provides functionality for writing initial Seed value to the channel to re-start CRC computation for the same channel configuration.

The `Crcu_SetChannelSeed()` function writes initial Seed value.

Note

In case when CRCU channel is configured for 16 bit CRC, only the 16 LSB bits are significative.

3.6.6 CRC Unit Synchronous Channel CRC Calculation

CRCU driver provides functionality for computing the CRC of a data stream of a given length. The `Crcu_SyncCalculateChannelCrc()` function is responsible for synchronous channel CRC calculation. The CRC computation uses the CRC width, polynom and swap/inversion options previously set for the channel by the function `Crcu_SetChannelConfig()` and the seed previously set for the channel by the function `Crcu_SetChannelSeed()`. The CRC is computed over the input daca vector pointed by `Crcu_DataPtr` and having the length set to `Crcu_Length`. The function performs the CRC calculation synchronously and returns the result CRC.

3.6.7 CRC Unit Asynchronous Channel CRC Calculation

The `Crcu_AsyncCalculateChannelCrc()` function is responsible for asynchronous channel CRC calculation. The CRC computation uses the CRC width, polynom and swap/inversion options previously set for the channel by the function `Crcu_SetChannelConfig()` and the seed previously set for the channel by the function `Crcu_SetChannelSeed()`. The CRC is computed over the input daca vector pointed by `Crcu_DataPtr` and having the length set to `Crcu_Length`. The function performs the CRC calculation asynchronously, by configuring and starting the DMA channel selected from configuration phase and does not return anything. When the DMA finishes sending the data to the CRC engine, the notification selected from configuration phase is called having the result CRC as parameter.

3.6.8 CRC Unit Channel Reading CRC Signature

Driver provides functionality for reading CRC signature from required channel.

The `Crcu_GetChannelCrc()` function reads CRC checksum.

3.6.9 CRCUExamples for using the driver

This sub-chapter lists some examples of how to use the CRCU Driver.

3.6.9.1 CCITT-FALSE CRC16 Example

To compute CRC checksum according to CCITT-FALSE CRC16 standard, please follow the steps below:

1. In Tresos GUI, add one configuration for the CRCU driver.
2. Create one CRCU channel in TRESOS CrcuChannel container.
3. Create one CRCU channel configuration in TRESOS CrcuChannelConfig, using the following values for the attributes:
   - Set `CRC Width` attribute to `CRCU_WIDTH_16`.
   - Set `Polynom` attribute to `0x1021`.
   - Set `Input Data Swap` attribute to `CRCU_IN_BYTESWAP`.
   - Set `Output Data Swap` attribute to `CRCU_OUT_NOSWAP`.
   - Set `Output Data Inversion` attribute to `CRCU_OUT_NOINVERSION`.
4. Generate the configuration files from Tresos.
5. Write the application code, following the steps:
   - Call the function `Crcu_Init()` providing it as parameter a NULL pointer.
   - Call the function `Crcu_SetChannelConfig()` providing it as parameters the symbolic names of the CRCU channel and CRCU channel configurations as they appear in the Tresos generated CDD_Crcu_Cfg.h file.
   - Call the function `Crcu_SetChannelSeed()` providing it as channel parameter the symbolic name of the CRCU channel generated in CDD_Crcu_Cfg.h and as seed parameter the value `0xFFFF`.
   - Call the function `Crcu_SyncCalculateChannelCrc()` providing it as channel parameter the symbolic name of the CRCU channel generated in CDD_Crcu_Cfg.h, the pointer to the data array to perform the CRC on and the length of this array. The return value of the function represents the CRC result.

3.6.9.2 CRC-32 IEEE 802.3 Example

To compute CRC checksum according to the IEEE 802.3 Ethernet standard, please follow the steps below:

1. In Tresos GUI, add one configuration for the CRCU driver.
2. Create one CRCU channel in TRESOS CrcuChannel container.

3. Create one CRCU channel configuration in TRESOS CrcuChannelConfig, using the following values for the attributes:
   - Set `CRC Width` attribute to `CRCU_WIDTH_32`.
   - Set `Polynom` attribute to `0x04C11DB7`.
   - Set `Input Data Swap` attribute to `CRCU_IN_BITANDBYTESWAP`.
   - Set `Output Data Swap` attribute to `CRCU_OUT_BITANDBYTESWAP`.
   - Set `Output Data Inversion` attribute to `CRCU_OUT_INVERSION`.
4. Generate the configuration files from Tresos.
5. Write the application code, following the steps:
   - Call the function `Crcu_Init()` providing it as parameter a NULL pointer.
   - Call the function `Crcu_SetChannelConfig()` providing it as parameters the symbolic names of the CRCU channel and CRCU channel configurations as they appear in the Tresos generated CDD_Crcu_Cfg.h file.
   - Call the function `Crcu_SetChannelSeed()` providing it as channel parameter the symbolic name of the CRCU channel generated in CDD_Crcu_Cfg.h and as seed parameter the value `0xFFFFFFFF`.
   - Call the function `Crcu_SyncCalculateChannelCrc()` providing it as channel parameter the symbolic name of the CRCU channel generated in CDD_Crcu_Cfg.h, the pointer to the data array to perform the CRC on and the length of this array. The return value of the function represents the CRC result.

3.7 Runtime Errors

The CRCU driver does not report any DEM error.

If development errors are enabled (CRCU_DEV_ERROR_DETECT is STD_ON), the following DET errors will be raised:

Table 3-2. Development Errors

Function	Error Code	Condition triggering the error
Crcu_SetChannelConfig	CRCU_E_UNINIT	API is called when the driver is not yet initialized.
Crcu_SetChannelConfig	CRCU_E_INVALID_CHANNEL	API is called with invalid channel ID parameter.
Crcu_SetChannelConfig	CRCU_E_INVALID_CHANNEL_CONFIG	API is called with invalid channel configuration ID parameter.
Crcu_SetChannelConfig	CRCU_E_CHANNEL_BUSY	API is called when other channel function is already processing on the same channel.
Crcu_SetChannelConfig	CRCU_E_INVALID_POINTER	API is called with a NULL pointer as parameter.
Crcu_SetChannelSeed	CRCU_E_UNINIT	API is called when the driver is not yet initialized.
Crcu_SetChannelSeed	CRCU_E_INVALID_CHANNEL	API is called with invalid channel ID parameter.

Table continues on the next page...

Table 3-2. Development Errors (continued)

Function	Error Code	Condition triggering the error
Crcu_SetChannelSeed	CRCU_E_CHANNEL_BUSY	API is called when other channel function is already processing on the same channel.
Crcu_SyncCalculateChannelCrc	CRCU_E_UNINIT	API is called when the driver is not yet initialized.
Crcu_SyncCalculateChannelCrc	CRCU_E_INVALID_CHANNEL	API is called with invalid channel ID parameter.
Crcu_SyncCalculateChannelCrc	CRCU_E_CHANNEL_BUSY	API is called when other channel function is already processing on the same channel.
Crcu_SyncCalculateChannelCrc	CRCU_E_INVALID_POINTER	API is called with a NULL pointer as parameter.
Crcu_SyncCalculateChannelCrc	CRCU_E_INVALID_LENGTH	API is called with length parameter set to 0.
Crcu_AsyncCalculateChannelCrc	CRCU_E_UNINIT	API is called when the driver is not yet initialized.
Crcu_AsyncCalculateChannelCrc	CRCU_E_INVALID_CHANNEL	API is called with invalid channel ID parameter.
Crcu_AsyncCalculateChannelCrc	CRCU_E_CHANNEL_BUSY	API is called when other channel function is already processing on the same channel.
Crcu_AsyncCalculateChannelCrc	CRCU_E_INVALID_POINTER	API is called with a NULL pointer as parameter.
Crcu_AsyncCalculateChannelCrc	CRCU_E_INVALID_LENGTH	API is called with length parameter set to 0.
Crcu_AsyncCalculateChannelCrc	CRCU_E_DMA_CH_NOT_CONFIGURED	API is called with for a channel that does not have a DMA channel configured.
Crcu_GetChannelCrc	CRCU_E_UNINIT	API is called when the driver is not yet initialized.
Crcu_GetChannelCrc	CRCU_E_INVALID_CHANNEL	API is called with invalid channel ID parameter.
Crcu_GetChannelCrc	CRCU_E_CHANNEL_BUSY	API is called when other channel function is already processing on the same channel.
Crcu_GetVersionInfo	CRCU_E_INVALID_POINTER	API is called with a NULL pointer as parameter.

3.8 Software specification

The following sections contains driver software specifications.

3.8.1 Define Reference

This chapter describes the defines supported by the CRCU driver.

3.8.1.1 Define CRCU_INSTANCE_ID

ID of CRC Instance.

Details:

Parameter used when raising an error/exception

Table 3-3. Define CRCU_INSTANCE_ID Description

Name	CRCU_INSTANCE_ID
Initializer	(0U)

3.8.1.2 Define CRCU_INIT_ID

API service ID for Crcu_Init function.

Details:

Parameter used when raising an error/exception

Table 3-4. Define CRCU_INIT_ID Description

Name	CRCU_INIT_ID
Initializer	(1U)

3.8.1.3 Define CRCU_SETCHANNELCONFIG_ID

API service ID for Crcu_SetChannelConfig function.

Details:

Parameter used when raising an error/exception

Table 3-5. Define CRCU_SETCHANNELCONFIG_ID Description

Name	CRCU_SETCHANNELCONFIG_ID
Initializer	(2U)

3.8.1.4 Define CRCU_SETCHANNELSEED_ID

API service ID for Crcu_SetChannelSeed function.

Details:

Parameter used when raising an error/exception

Table 3-6. Define CRCU_SETCHANNELSEED_ID Description

Name	CRCU_SETCHANNELSEED_ID
Initializer	(3U)

3.8.1.5 Define CRCU_SYNCCALCULATECHANNELCRC_ID

API service ID for Crcu_SyncCalculateChannelCrc function.

Details:

Parameter used when raising an error/exception

Table 3-7. Define CRCU_SYNCCALCULATECHANNELCRC_ID Description

Name	CRCU_SYNCCALCULATECHANNELCRC_ID
Initializer	(4U)

3.8.1.6 Define CRCU_ASYNCCALCULATECHANNELCRC_ID

API service ID for Crcu_SyncCalculateChannelCrc function.

Details:

Parameter used when raising an error/exception

Table 3-8. Define CRCU_ASYNCCALCULATECHANNELCRC_ID Description

Name	CRCU_ASYNCCALCULATECHANNELCRC_ID
Initializer	(5U)

3.8.1.7 Define CRCU_GETCHANNELCRC_ID

API service ID for Crcu_GetChannelCrc function.

Details:

Parameter used when raising an error/exception

Table 3-9. Define CRCU_GETCHANNELCRC_ID Description

Name	CRCU_GETCHANNELCRC_ID
Initializer	(6U)

3.8.1.8 Define CRCU_GETVERSIONINFO_ID

API service ID for Crcu_GetVersionInfo function.

Details:

Parameter used when raising an error/exception

Table 3-10. Define CRCU_GETVERSIONINFO_ID Description

Name	CRCU_GETVERSIONINFO_ID
Initializer	(10U)

3.8.1.9 Define CRCU_E_UNINIT

API service is called before driver is initialized.

Details:

Parameter is used when raising a Det error

Table 3-11. Define CRCU_E_UNINIT Description

Name	CRCU_E_UNINIT
Initializer	(1U)

3.8.1.10 Define CRCU_E_INVALID_CHANNEL

API service is called with wrong channel identifier.

Details:

Parameter is used when raising a Det error

Table 3-12. Define CRCU_E_INVALID_CHANNEL Description

Name	CRCU_E_INVALID_CHANNEL
Initializer	(2U)

3.8.1.11 Define CRCU_E_INVALID_CHANNEL_CONFIG

API service is called with wrong channel configuration identifier.

Details:

Parameter is used when raising a Det error

Table 3-13. Define CRCU_E_INVALID_CHANNEL_CONFIG Description

Name	CRCU_E_INVALID_CHANNEL_CONFIG
Initializer	(3U)

3.8.1.12 Define CRCU_E_INVALID_POINTER

API service is called with NULL pointer parameter.

Details:

Parameter is used when raising a Det error

Table 3-14. Define CRCU_E_INVALID_POINTER Description

Name	CRCU_E_INVALID_POINTER
Initializer	(4U)

3.8.1.13 Define CRCU_E_INVALID_LENGTH

API service is called with length parameter set to zero.

Details:

Parameter is used when raising a Det error

Table 3-15. Define CRCU_E_INVALID_LENGTH Description

Name	CRCU_E_INVALID_LENGTH
Initializer	(5U)

3.8.1.14 Define CRCU_E_DMA_CH_NOT_CONFIGURED

Crcu_AsyncCalculateChannelCrc API service is called for a channel that does not have a DMA channel configured.

Details:

Parameter is used when raising a Det error

Table 3-16. Define CRCU_E_DMA_CH_NOT_CONFIGURED Description

Name	CRCU_E_DMA_CH_NOT_CONFIGURED
Initializer	(6U)

3.8.1.15 Define CRCU_E_CHANNEL_BUSY

API channel service is called while another channel service is running on the same channel.

Details:

Parameter is used when raising a Det error

Table 3-17. Define CRCU_E_CHANNEL_BUSY Description

Name	CRCU_E_CHANNEL_BUSY
Initializer	(7U)

3.8.1.16 Define CRCU_DEV_ERROR_DETECT

Enables/Disables Development Error Detection.

Table 3-18. Define CRCU_DEV_ERROR_DETECT Description

Name	CRCU_DEV_ERROR_DETECT
Initializer	(STD_ON)

3.8.1.17 Define CRCU_PRECOMPILE_SUPPORT

Crcu driver Pre-Compile configuration switch.

Table 3-19. Define CRCU_PRECOMPILE_SUPPORT Description

Name	CRCU_PRECOMPILE_SUPPORT
Initializer	(STD_ON)

3.8.1.18 Define CRCU_GET_VERSION_INFO_API

Enable/disable switch for function Crcu_GetVersionInfo().

Table 3-20. Define CRCU_GET_VERSION_INFO_API Description

Name	CRCU_GET_VERSION_INFO_API
Initializer	(STD_ON)

3.8.1.19 Define CRCU_DMA_USED

Enable/disable switch for DMA support in Crcu driver. Also enables/disables the presence of the function Crcu_AsyncCalculateChannelCrc() in code.

Table 3-21. Define CRCU_DMA_USED Description

Name	CRCU_DMA_USED
Initializer	(STD_OFF)

3.8.1.20 Define CrcuConf_CrcuChannel_CrcuChannel_0

Symbolic names for the Crcu channels.

Table 3-22. Define CrcuConf_CrcuChannel_CrcuChannel_0 Description

Name	CrcuConf_CrcuChannel_CrcuChannel_0
Initializer	(CRC_0)

3.8.1.21 Define CrcuConf_CrcuChannelConfig_CrcuChannelConfig_0

Symbolic names for the Crcu channel configurations.

Violates: The compiler/linker shall be checked to ensure that 31 character significance and case sensitivity are supported for external identifiers.

Table 3-23. Define CrcuConf_CrcuChannelConfig_CrcuChannelConfig_0 Description

Name	CrcuConf_CrcuChannelConfig_CrcuChannelConfig_0
Initializer	(0U)

3.8.2 Enum Reference

Enumeration of all constants supported by the driver are as per AUTOSAR CRCU Driver software specification Version 4.2 Rev0002 .

3.8.2.1 Structure Crcu_ChannelStateType

Structure needed by `Crcu_SetChannelConfig()`, `Crcu_SetChannelSeed()`, `Crcu_AsyncCalculateChannelCrc()`, `Crcu_GetChannelCrc()`.

Details:

The structure `Crcu_ChannelStateType` Enumerator that defines CRCU channel states.

Declaration:

```
typedef enum
{
    CRCU_CH_STATE_IDLE = 0U,
    CRCU_CH_STATE_BUSY
}
```

Table 3-24. Enumeration Crcu_ChannelStateType member description

Member	Description
CRCU_CH_STATE_IDLE	Channel is in IDLE state.
CRCU_CH_STATE_BUSY	Channel is in BUSY state.

3.8.3 Function Reference

This chapter describes the functions supported by the CRCU driver.

3.8.3.1 Function Crcu_Init

This function initializes the driver.

Details:

This service is a non reentrant function used for driver initialization.

Return: void.

Prototype: `void Crcu_Init(Crcu_ConfigType *Crcu_ConfigPtr);`

Table 3-25. Crcu_Init Arguments

Type	Name	Direction	Description
Crcu_ConfigType*	Crcu_ConfigPtr	Input	Pointer to a configuration structure.

3.8.3.2 Function Crcu_SetChannelConfig

This function configures a CRC channel.

Details:

This service is used for CRC channel configuration.

Return: Crcu_ChannelAddressType - Address of the CRC channel data register.

Prototype: `Crcu_ChannelAddressType Crcu_SetChannelConfig(Crcu_ChannelIdType Crcu_ChannelId, Crcu_ChannelConfigIdType Crcu_ChannelConfigId);`

Table 3-26. Crcu_SetChannelConfig Arguments

Type	Name	Direction	Description
Crcu_ChannelIdType	Crcu_ChannelId	Input	CRCU channel identifier.
Crcu_ChannelConfigIdType	Crcu_ChannelConfigId	Input	CRCU channel configuration identifier.

3.8.3.3 Function Crcu_SetChannelSeed

This function writes a seed value to a CRC channel.

Details:

The function writes the seed value of a CRC channel.

Return: void.

Prototype: `void Crcu_SetChannelSeed(Crcu_ChannelIdType Crcu_ChannelId, Crcu_ValueType Crcu_Seed);`

Table 3-27. Crcu_SetChannelSeed Arguments

Type	Name	Direction	Description
Crcu_ChannelIdType	Crcu_ChannelId	Input	CRCU channel identifier.
Crcu_ValueType	Crcu_Seed	Input	Value of the seed.

3.8.3.4 Function Crcu_SyncCalculateChannelCrc

This function calculates CRC synchronously and returns it immediately.

Details:

The function calculates and returns the CRC for a previously configured channel.

Return: Crcu_ValueType - value of the CRC.

Prototype: `Crcu_ValueType Crcu_SyncCalculateChannelCrc(Crcu_ChannelIdType Crcu_ChannelId, uint8 *Crcu_DataPtr, uint32 Crcu_Length);`

Table 3-28. Crcu_SyncCalculateChannelCrc Arguments

Type	Name	Direction	Description
Crcu_ChannelIdType	Crcu_ChannelId	Input	CRCU channel identifier.
uint8 *	Crcu_DataPtr	Input	Pointer to data to perform CRC on.
uint32	Crcu_Length	Input	Length of data to perform CRC on, in bytes.

3.8.3.5 Function Crcu_AsyncCalculateChannelCrc

This function initiates the calculation of the CRC with the help of a DMA channel and returns immediatelly without returning any value. The value of the CRC will be received by the caller through the CRC notification function configured for the used channel.

Details:

The function initiates the calculation of the CRC for a previously configured channel.

Return: void

Prototype: `void Crcu_AsyncCalculateChannelCrc(Crcu_ChannelIdType Crcu_ChannelId, uint8 *Crcu_DataPtr, uint32 Crcu_Length);`

Table 3-29. Crcu_AsyncCalculateChannelCrc Arguments

Type	Name	Direction	Description
Crcu_ChannelIdType	Crcu_ChannelId	Input	CRCU channel identifier.
uint8 *	Crcu_DataPtr	Input	Pointer to data to perform CRC on.
uint32	Crcu_Length	Input	Length of data to perform CRC on, in bytes.

3.8.3.6 Function Crcu_GetChannelCrc

Provides CRC result.

Details:

Returns CRC result for required channel.

Return: Crcu_ValueType - CRC result.

Prototype: `Crcu_ValueType Crcu_GetChannelCrc(Crcu_ChannelIdType Crcu_ChannelId);`

Table 3-30. Crcu_GetChannelCrc Arguments

Type	Name	Direction	Description
Crcu_ChannelIdType	Crcu_ChannelId	Input	CRCU channel identifier.

3.8.3.7 Function Crcu_GetVersionInfo

Software module version query.

Details:

Returns the version information of this module

Return: void.

Prototype: `void Crcu_GetVersionInfo(Std_VersionInfoType *Crcu_VersionInfoPtr);`

Table 3-31. Crcu_GetVersionInfo Arguments

Type	Name	Direction	Description
Std_VersionInfoType *	Crcu_VersionInfoPtr	Output	Pointer to address where version information will be copied.

3.8.4 Structs Reference

This chapter describes the structs supported by the CRCU driver.

3.8.4.1 Structure Crcu_ChannelType

Structure that defines a CRCU channel.

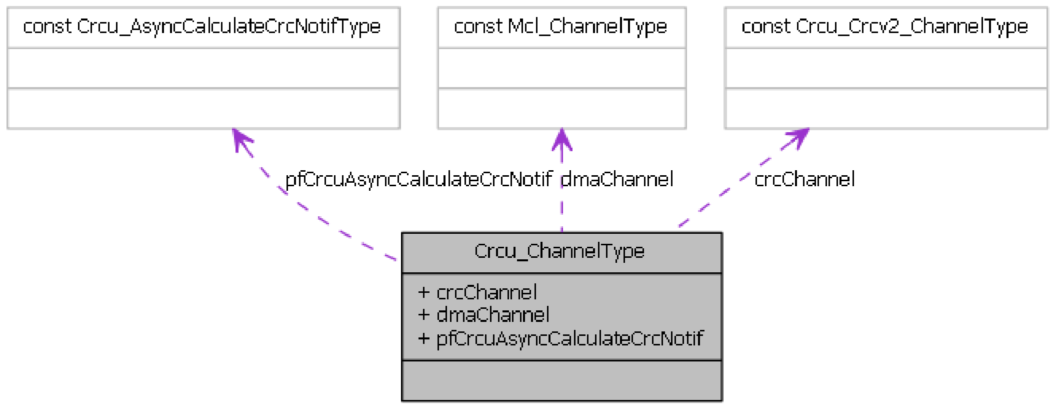

const Crcu_AsyncCalculateCrcNotifType
const Mcl_ChannelType
const Crcu_Crcv2_ChannelType
pfCrcuAsyncCalculateCrcNotif
dmaChannel
crcChannel
Crcu_ChannelType
+ crcChannel
+ dmaChannel
+ pfCrcuAsyncCalculateCrcNotif

Figure 3-1. Struct Crcu_ChannelType

Declaration:

```
typedef struct
                           {
                             constCrcu_Crcv2_ChannelType crcChannel,
                                     const Mcl_ChannelType dmaChannel,
                                     constCrcu_AsyncCalculateCrcNotifType
pfCrcuAsyncCalculateCrcNotif
                           } Crcu_ChannelType;
```

Table 3-32. Structure Crcu_ChannelType member description

Member	Description
crcChannel	Crc channel assigned to the CRCU logical channel.
dmaChannel	Dma channel assigned to the CRCU logical channel.
pfCrcuAsyncCalculateCrcNotif	Notification for completion of async crc calculation.

3.8.4.2 Structure Crcu_ConfigType

Structure that defines the format of the Crcu driver configuration data.

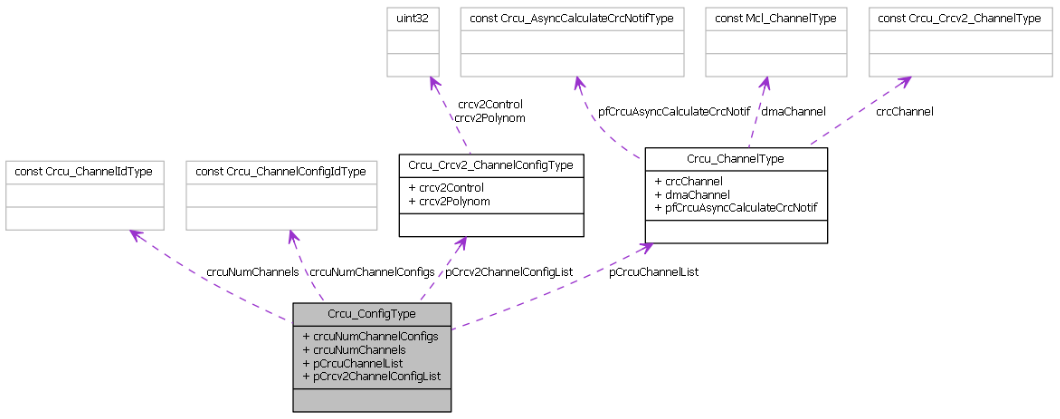


Figure 3-2. Struct Crcu_ConfigType

Implements: Crcu_ConfigType_struct

Declaration:

```
typedef struct
                           {
                             constCrcu_ChannelConfigIdType crcuNumChannelConfigs,
                                     constCrcu_ChannelIdType crcuNumChannels,
                                     constCrcu_ChannelType(* pCrcuChannelList)[],
                                     constCrcu_Crcv2_ChannelConfigType(*
pCrcv2ChannelConfigList)[]
                           } Crcu_ConfigType;
```

Table 3-33. Structure Crcu_ConfigType member description

Member	Description
crcuNumChannelConfigs	Number of Crcu channel configurations.
crcuNumChannels	Number of Crcu channels.
pCrcuChannelList	List of Crcu channels.
pCrcv2ChannelConfigList	List of Crcu channel configurations.

3.8.5 Types Reference

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

3.8.5.1 Typedef Crcu_ChannelIdType

Type that defines the identifier of a CRCU channel.

Details:

Provides CRCU channel selection

Implements: Crcu_ChannelIdType_typedef

Type: uint8

3.8.5.2 Typedef Crcu_ChannelConfigIdType

Type that defines the identifier of a CRCU channel configuration.

Details:

Provides CRCU channel configuration selection

Implements: Crcu_ChannelConfigIdType_typedef

Type: uint8

3.8.5.3 Typedef Crcu_ChannelAddressType

Crcu_ChannelAddressType.

Details:

Type for abstracting the address of the CRC feeding register

Implements: Crcu_ChannelAddressType_typedef

Type:`Crcu_Crcv2_ChannelAddressType`

3.8.5.4 Typedef Crcu_ValueType

Crcu_ValueType.

Details:

Type for abstracting the CRC computation values

Implements: Crcu_ValueType_typedef

Type:`Crcu_Crcv2_ValueType`

3.8.5.5 Typedef Crcu_AsyncCalculateCrcNotifType

Crcu async calculate Crc complete notification type. The callback notification shall be configurable as pointer to user defined function within the configuration structure.

Type:typedef P2FUNC(void, CRCU_APPL_CODE, Crcu_AsyncCalculateCrcNotifType) (VAR(Crcu_ChannelIdType, AUTOMATIC) channelId, VAR(Crcu_ValueType, AUTOMATIC) crc)

3.9 Symbolic Names DISCLAIMER

All containers having the symbolic name tag set as true in the Autosar schema will generate defines like:

#define <Container_Short_Name> <Container_ID>

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

Chapter 4
Tresos Configuration Plug-in

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

4.1 Configuration elements of Crcu

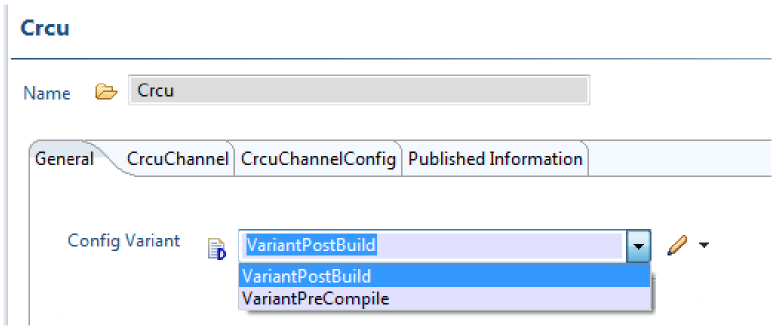
Included forms :

- IMPLEMENTATION_CONFIG_VARIANT
- CrcuGeneral
- CrcuConfig
- CommonPublishedInformation

4.2 Form IMPLEMENTATION_CONFIG_VARIANT

VariantPreCompile: Only precompile time configuration parameters. Only one set of parameters.

VariantPostBuild: Mix of precompile and postbuild time configuration parameters.



Name: Crcu

Config Variant: VariantPostBuild

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

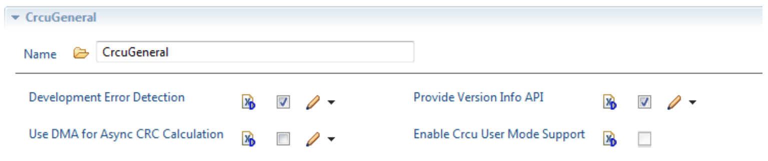
Table 4-1. Attribute IMPLEMENTATION_CONFIG_VARIANT detailed description

Property	Value
Label	Config Variant
Type	ENUMERATION
Default	VariantPostBuild
Range	VariantPostBuild VariantPreCompile

4.3 Form CrcuGeneral

CrcuGeneral

All general parameters of the Crcu driver are collected here.



CrcuGeneral
Name
CrcuGeneral
Development Error Detection
Provide Version Info API
Use DMA for Async CRC Calculation
Enable Crcu User Mode Support

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

4.3.1 CrcuDevErrorDetect (CrcuGeneral)

Crcu Development Error Detect

Compile switch to enable / disable development error detection for this module.

- Unchecked: Development error detection disabled
- Checked : Development error detection enabled

Table 4-2. Attribute CrcuDevErrorDetect (CrcuGeneral) detailed description

Property	Value
Label	Development Error Detection
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	true

4.3.2 CrcuVersionInfoApi (CrcuGeneral)

Crcu VersionInfo Api

Compile switch to enable / disable the version information API.

- Checked : API enabled
- Unchecked:API disabled

Table 4-3. Attribute CrcuVersionInfoApi (CrcuGeneral) detailed description

Property	Value
Label	Provide Version Info API
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	true

4.3.3 CrcuDmaUsed (CrcuGeneral)

Check this in order to be able to use DMA in the Crcu driver.

Leaving this unchecked will allow the Crcu driver to compile with no dependencies from the Mcl driver.

Note: Implementation Specific Parameter.

Table 4-4. Attribute CrcuDmaUsed (CrcuGeneral) detailed description

Property	Value
Label	Use DMA for Async CRC Calculation
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

4.3.4 CrcuEnableUserModeSupport (CrcuGeneral)

This parameter is added in Crcu configuration in order to keep a consistent design over the entire set of MCAL drivers. It cannot be configured by the user and is always set to 'false'. There are no registers used by the driver which require special measures in order to be accessed from user mode, so Crcu driver can be run from either user or supervisor mode.

Table 4-5. Attribute CrcuEnableUserModeSupport (CrcuGeneral) detailed description

Property	Value
Label	Enable Crcu User Mode Support
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

4.4 Form CrcuConfig

Included forms :

- Form CrcuChannel
- Form CrcuChannelConfig

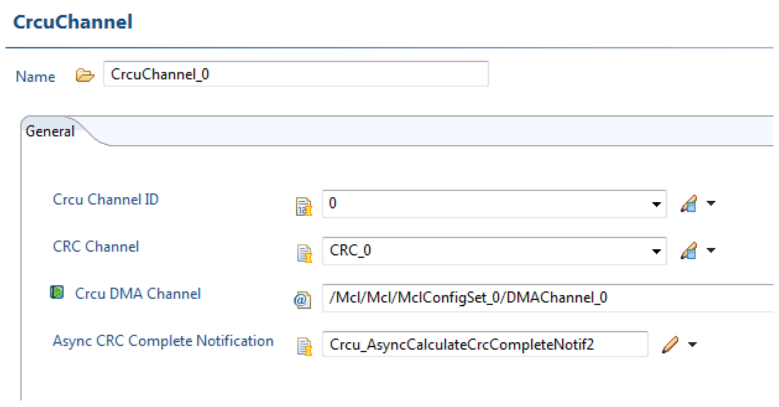
Name: CrcuConfig

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

4.4.1 Form CrcuChannel

Configuration of an individual Crcu channel. Symbolic names will be generated for each channel.

Is included by form : Form CrcuConfig


CrcuChannel

Name: CrcuChannel_0

General

Crcu Channel ID: 0

CRC Channel: CRC_0

Crcu DMA Channel: /Mcl/Mcl/MclConfigSet_0/DMAChannel_0

Async CRC Complete Notification: Crcu_AsyncCalculateCrcCompleteNotif2

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

4.4.1.1 CrcuChannelId (CrcuChannel)

Identifies the Crcu channel.

Note: Implementation Specific Parameter.

Table 4-6. Attribute CrcuChannelId (CrcuChannel) detailed description

Property	Value
Label	Crcu Channel ID
Type	INTEGER
Origin	NXP
Symbolic Name	true
Invalid	Range >=0 <=255

4.4.1.2 CrcChannel (CrcuChannel)

Selects one of the CRC channels available on the platform.

Table 4-7. Attribute CrcChannel (CrcuChannel) detailed description

Property	Value
Label	CRC Channel
Type	ENUMERATION
Origin	NXP
Symbolic Name	false

4.4.1.3 CrcuAsyncCalculateCrcCompleteNotification (CrcuChannel)

Function name of notification called when an async CRC calculation operation has completed.

Note: Implementation Specific Parameter.

Table 4-8. Attribute CrcuAsyncCalculateCrcCompleteNotification (CrcuChannel) detailed description

Property	Value
Label	Async CRC Complete Notification
Type	FUNCTION-NAME
Origin	NXP
Symbolic Name	false
Default	Crcu_AsyncCalculateCrcCompleteNotif

4.4.1.4 CrcuDmaChannelRef (CrcuChannel)

Reference to a DMA channel (set in the Mcl driver configuration) used by the Crcu driver when calling function Crcu_AsyncCalculateCrc().

Note: Implementation Specific Parameter.

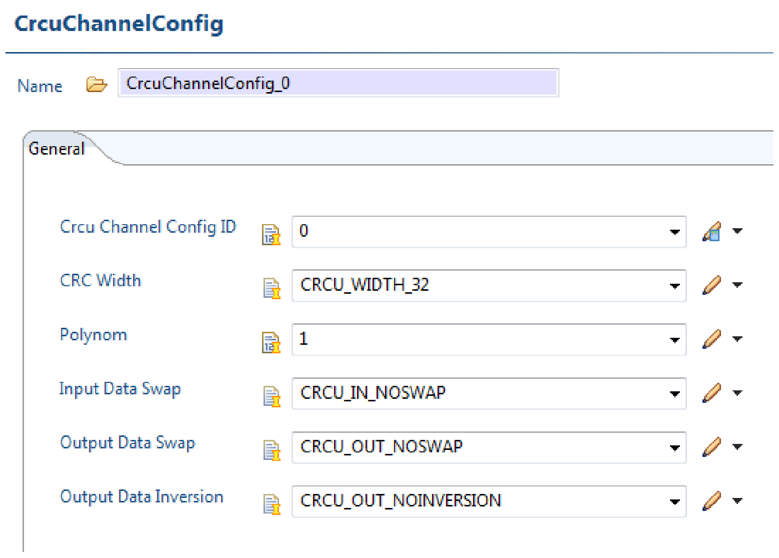
Table 4-9. Attribute CrcuDmaChannelRef (CrcuChannel) detailed description

Property	Value
Label	Crcu DMA Channel
Type	REFERENCE
Origin	Custom

4.4.2 Form CrcuChannelConfig

A Crcu channel configuration represents a group of settings that define the way the CRC is going to be computed.

Is included by form : Form CrcuConfig


CrcuChannelConfig

Name	CrcuChannelConfig_0

General

Crcu Channel Config ID	0
CRC Width	CRCU_WIDTH_32
Polynom	1
Input Data Swap	CRCU_IN_NOSWAP
Output Data Swap	CRCU_OUT_NOSWAP
Output Data Inversion	CRCU_OUT_NOINVERSION

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

4.4.2.1 CrcuChannelConfigId (CrcuChannelConfig)

Identifies the Crcu channel configuration.

Note: Implementation Specific Parameter.

Table 4-10. Attribute CrcuChannelConfigId (CrcuChannelConfig) detailed description

Property	Value
Label	Crcu Channel Config ID
Type	INTEGER
Origin	NXP
Symbolic Name	true
Invalid	Range >=0 <=255

4.4.2.2 Crcu_Width (CrcuChannelConfig)

Width of the CRC protocol to be used. (16, 32 bit)

Table 4-11. Attribute Crcu_Width (CrcuChannelConfig) detailed description

Property	Value
Label	CRC Width

Table continues on the next page...

Table 4-11. Attribute Crcu_Width (CrcuChannelConfig) detailed description (continued)

Property	Value
Type	ENUMERATION
Origin	NXP
Symbolic Name	false
Default	CRCU_WIDTH_32
Range	CRCU_WIDTH_16 CRCU_WIDTH_32

4.4.2.3 Crcu_Polynom (CrcuChannelConfig)

The polynom that will be used during CRC calculation

Table 4-12. Attribute Crcu_Polynom (CrcuChannelConfig) detailed description

Property	Value
Label	Polynom
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	1
Invalid	Range >=1 <=4294967295

4.4.2.4 Crcu_In_Swap (CrcuChannelConfig)

Enumerator that defines CRC input data bitwise and bytewise swap (transpose) functionality.

Table 4-13. Attribute Crcu_In_Swap (CrcuChannelConfig) detailed description

Property	Value
Label	Input Data Swap
Type	ENUMERATION
Origin	NXP
Symbolic Name	false
Default	CRCU_IN_NOSWAP
Range	CRCU_IN_NOSWAP CRCU_IN_BITSWAP CRCU_IN_BITANDBYTESWAP CRCU_IN_BYTESWAP

4.4.2.5 Crcu_Out_Swap (CrcuChannelConfig)

Enumerator that defines CRC output data bitwise and bytewise swap (transpose) functionality.

Table 4-14. Attribute Crcu_Out_Swap (CrcuChannelConfig) detailed description

Property	Value
Label	Output Data Swap
Type	ENUMERATION
Origin	NXP
Symbolic Name	false
Default	CRCU_OUT_NOSWAP
Range	CRCU_OUT_NOSWAP CRCU_OUT_BITSWAP CRCU_OUT_BITANDBYTESWAP CRCU_OUT_BYTESWAP

4.4.2.6 Crcu_Out_Inversion (CrcuChannelConfig)

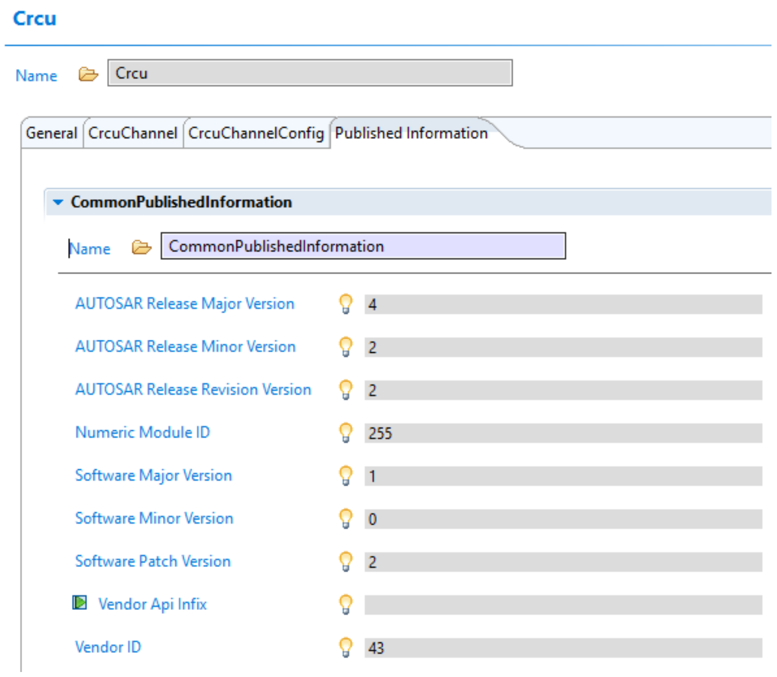
Enumerator that defines CRC output Inversion functionality.

Table 4-15. Attribute Crcu_Out_Inversion (CrcuChannelConfig) detailed description

Property	Value
Label	Output Data Inversion
Type	ENUMERATION
Origin	NXP
Symbolic Name	false
Default	CRCU_OUT_NOINVERSION
Range	CRCU_OUT_NOINVERSION CRCU_OUT_INVERSION

4.5 Form CommonPublishedInformation

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


Crcu

Name: Crcu

General | CrcuChannel | CrcuChannelConfig | Published Information

CommonPublishedInformation

Name: CommonPublishedInformation

AUTOSAR Release Major Version	4
AUTOSAR Release Minor Version	2
AUTOSAR Release Revision Version	2
Numeric Module ID	255
Software Major Version	1
Software Minor Version	0
Software Patch Version	2
Vendor Api Infix	
Vendor ID	43

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

4.5.1 ArReleaseMajorVersion (CommonPublishedInformation)

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

Table 4-16. Attribute ArReleaseMajorVersion (CommonPublishedInformation) detailed description

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

4.5.2 ArReleaseMinorVersion (CommonPublishedInformation)

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

Table 4-17. Attribute ArReleaseMinorVersion (CommonPublishedInformation) detailed description

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

4.5.3 ArReleaseRevisionVersion (CommonPublishedInformation)

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

Table 4-18. Attribute ArReleaseRevisionVersion (CommonPublishedInformation) detailed description

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

4.5.4 ModuleId (CommonPublishedInformation)

Module ID of this module from Module List.

Table 4-19. Attribute ModuleId (CommonPublishedInformation) detailed description

Property	Value
Label	Module Id
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false

Table continues on the next page...

Table 4-19. Attribute ModuleId (CommonPublishedInformation) detailed description (continued)

Property	Value
Default	255
Invalid	Range >=255 <=255

4.5.5 SwMajorVersion (CommonPublishedInformation)

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

Table 4-20. Attribute SwMajorVersion (CommonPublishedInformation) detailed description

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

4.5.6 SwMinorVersion (CommonPublishedInformation)

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

Table 4-21. Attribute SwMinorVersion (CommonPublishedInformation) detailed description

Property	Value
Label	Software Minor Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	0
Invalid	Range >=0 <=0

4.5.7 SwPatchVersion (CommonPublishedInformation)

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

Table 4-22. Attribute SwPatchVersion (CommonPublishedInformation) detailed description

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

4.5.8 VendorApiInfix (CommonPublishedInformation)

In driver modules which can be instantiated several times on a single ECU, BSW00347 requires that the name of APIs is extended by the VendorId and a vendor specific name. This parameter is used to specify the vendor specific name. In total, the implementation specific name is generated as follows: <ModuleName>_>VendorId>_<VendorApiInfix><Api name from SWS>. E.g. assuming that the VendorId of the implementor is 123 and the implementer chose a VendorApiInfix of "v11r456" a api name Can_Write defined in the SWS will translate to Can_123_v11r456Write. This parameter is mandatory for all modules with upper multiplicity > 1. It shall not be used for modules with upper multiplicity =1.

Table 4-23. Attribute VendorApiInfix (CommonPublishedInformation) detailed description

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

4.5.9 VendorId (CommonPublishedInformation)

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

Table 4-24. Attribute VendorId (CommonPublishedInformation) detailed description

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

Document Number UM2CRCUASR4.2 Rev0002R1.0.2
Revision 1.0