User Manual

for MPC574XG FR Driver

Document Number: UM35FRASR4.2 Rev002R1.0.0
Rev. 1.0.0

Contents

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

Chapter 1 Revision History

Chapter 2 Introduction

Chapter 3 Driver

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

Chapter 4
Tresos Configuration Plug-in

Section number | Title | Page

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0.0	09-Feb-2017	Nam Khuc	RTM 1.0.0 version

Chapter 2
Introduction

This User’s Manual describes NXP Semiconductor AUTOSAR FlexRay driver.

AUTOSAR FR driver configuration parameters description can be found in Configuration Parameters section. Deviations from the specification are described in the Deviations from Requirements section. AUTOSAR FR driver requirements and APIs are described in the AUTOSAR 4.2 Rev0002FR Driver Software Specification Document [Table 2-3] and in API Reference section.

2.1 Supported Derivatives

The software described in this document is intended to be used with the following microcontroller devices of NXP Semiconductor:

Table 2-1. MPC574XG Derivatives

NXP Semiconductor	MPC5748G_LQFP176, MPC5748G_MAPBGA256, MPC5748G_MAPBGA324, MPC5747G_LQFP176, MPC5747G_MAPBGA256, MPC5747G_MAPBGA324, MPC5746G_LQFP176, MPC5746G_MAPBGA256, MPC5746G_MAPBGA324, MPC5748C_LQFP176, MPC5748C_MAPBGA256, MPC5748C_MAPBGA324, MPC5747C_LQFP176, MPC5747C_MAPBGA256, MPC5747C_MAPBGA324, MPC5746C_LQFP176, MPC5746C_MAPBGA256, MPC5746C_MAPBGA324, MPC5746C_MAPBGA100, MPC5745C_LQFP176, MPC5745C_MAPBGA256, MPC5745C_MAPBGA100, MPC5744C_LQFP176, MPC5744C_MAPBGA256,

Table 2-1. MPC574XG Derivatives

	MPC5744C_MAPBGA100, MPC5746B_LQFP176, MPC5746B_MAPBGA256, MPC5746B_MAPBGA100, MPC5744B_LQFP176, MPC5744B_MAPBGA256, MPC5744B_MAPBGA100, MPC5745B_LQFP176, MPC5745B_MAPBGA256, MPC5745B_MAPBGA100

All of the above microcontroller devices are collectively named as MPC574XG.

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
FR	FlexRay
FRIF	FlexRay Interface
BSW	Basic Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
ISR	Interrupt Service Routine
OS	Operating System
GUI	Graphical User Interface
API	Application Programming Interface
PB Variant	Post Build Variant
LT Variant	Link Time Variant
PC Variant	Pre Compile Variant

2.5 Reference List

Table 2-3. Reference List

#	Title	Version
1	AUTOSAR 4.2 Rev0002FR Driver Software Specification Document.	2.6.0 R4.2.1 Rev 1
2	MPC5748G Reference Manual	Rev. 5, 12/2016
3	MPC5746C Reference Manual	Rev. 4, 12/2016
4	MPC5748G_1N81M_Rev.2 (official document) (1N81M)	Jun-16
5	MPC5748G_1N81M_0N78S_Comparison_Summary_v2_0 (internal document) (1N81M, 0N78S)	31.10.2016
6	MPC5746C_1N06M_Rev.4 (official document) (1N06M)	Jul-16
7	MPC5746C_cut1.1_cut2.0_cut2.1_comparison_v0 (internal document) (1N06M, 0N84S, 1N84S)	14-Sep-16
8	C3M_cut2.1_new_errata_20170113 (internal document) (1N84S)	13-Jan-17

Chapter 3
Driver

3.1 Driver Design Summary

The driver provides services for the following (FlexRay controller) features:

- Stores configuration data address to enable subsequent API calls to access the configuration data
- Configures all FlexRay cluster and node configuration parameters
- Provides set of API functions which allow to start communication or stop communication (immediately or at the end of the current FlexRay communication cycle)
- Data transmission and reception in both static and dynamic parts of communication cycle is serviced by several Fr module API functions
- Protocol state machine control of the FlexRay CC is implemented by several Fr module API functions
- Provides information about FlexRay network global time
- Evaluates whether the CC is synchronized to the global FlexRay time
- API functions for configuration of wakeup channel and sending of the wakeup pattern are implemented
- Individual message buffer reconfiguration is supported
- Support of two independent receive FIFOs (one receive FIFO per channel)
- Driver provides set of API functions to support absolute timer
- Driver provides set of API functions for timer interrupt servicing
- Driver provides API for obtaining the node and cluster configuration parameters from the FlexRay CC Protocol Configuration Registers
- Driver provides DEM and DET error notification

3.2 Deviations from Requirements

The driver deviates from the AUTOSAR FR Driver software specification in some places. Table 3-2 identifies the AUTOSAR requirements that are not fully implemented, implemented differently, or out of scope for the FR driver. Table 3-1 provides Status column description.

Table 3-1. Deviations Status Column Description

Term	Definition
N/A	Not available
N/T	Not testable
N/S	Out of scope
N/I	Not implemented
N/F	Not fully implemented

Table 3-2. Requirements Deviations

Requirement	Status	Description	Notes
FR073	N/F	Conforms to C programming language in conformance to the HIS subset of the MISRA C Standard.	FR driver was designed to comply with MISRA C Standard. All violations against MISRA C Standard are documented and explained in the AUTOSAR_MCAL_FR_MisraReport.xlsx document.
FR076	N/I	Replaces all prefixes Fr within this specification by a vendor specific prefix Fr_<Vendor Id>_<Vendor specific name> for implementation to allow usage of different FlexRay Drivers within one build system. This rule applies to all prefixes in filenames, Fr module specific datatypes, Fr module specific constants, Fr module specific global variables, API functions and DEM event Ids.	The driver does not support a vendor specific prefix.
FR106	N/S	The Fr module and/or the underlying hardware shall stop FlexRay timers in case of loss of synchronization.	Supported directly by the NXP Semiconductor FlexRay hardware.
FR111	N/S	The module source code files shall include SchM_Fr.h if data consistency mechanisms of the BSW scheduler are required.	Data consistency mechanisms of the BSW scheduler are not required.
FR391	N/I	Optional interfaces `SchM_Exit_Fr` and `SchM_Enter_Fr` are required to fulfill an optional functionality of the module.	`SchM_Exit_Fr` and `SchM_Enter_Fr` are not implemented.
FR606	N/F	If the optional configuration parameter `FrIfDemFTSlotStatusRef` exists and a single slot status error bit (vSS!SyntaxError, vSS!ContentError, vSS!Bviolation) is set, then the slot status information shall be reported to DEM as Dem_ReportErrorStatus (`FrIfDemFTSlotStatusRef`, DEM_EVENT_STATUS_FAILED).	These slot status bits are not related to the transmit message buffers for FlexRay Communications System Protocol Specification, Version 2.1 Rev A compliant controllers. Only vSS!TxConflict error bit is directly related to the transmission process.

3.3 Runtime Errors

3.3.1 Development Error Tracer Description

The driver generates the following DET errors at runtime.

Table 3-3. DET errors description

Name	Condition	Suggestion
FR_E_INV_TIMER_IDX_U8	An attempt to configure an invalid timer, Fr_AbsTimerIdx has an invalid value	Check the parameter Fr_AbsTimerIdx
FR_E_PARAM_POINTER	Invalid pointer in parameter list, pointer equals NULL_PTR	Check the pointer
FR_E_INV_OFFSET_U8	An attempt to configure timer for invalid macrotick offset number, Fr_Offset has an invalid value	Check the parameter Fr_Offset
FR_E_INV_CTRL_IDX_U8	An attempt to configure unsupported CC, Fr_CtrlIdx has an invalid value	Check the parameter Fr_CtrlIdx
FR_E_INV_CHNL_IDX_U8	Fr_ChnlIdx has an invalid value	Check the parameter Fr_ChnlIdx
FR_E_INV_CYCLE_U8	Invalid parameter cycle number	Check the parameter Fr_Cycle or Fr_CycleRepetition or Fr_CycleOffset
FR_E_INIT_FAILED	The Fr was not initialized successfully prior to this API function call	Call function Fr_Init() first
FR_E_INV_POCSTATE_U8	Fr CC is not in the expected POC state	Check the current POC state
FR_E_INV_LENGTH_U8	Payload length parameter has an invalid value	Check payload length paramter
FR_E_INV_LPDU_IDX_U8	Fr_LPduIdx has an invalid value	Check the parameter Fr_LPduIdx
FR_E_INV_HEADERCRC_U8	Fr_HeaderCRC has an invalid value for static segment	Check the parameter Fr_HeaderCRC
FR_E_INV_CONFIG_IDX_U8	Fr_ConfigParamIdx has an invalid value	Check the parameter Fr_ConfigParamIdx
FR_E_INV_FRAMELIST_SIZE	The parameter Fr_ListSize is larger than 15	Check the parameter Fr_ListSize

3.3.2 Diagnostic Event Manager Description

The driver generates the following DEM errors at runtime.

Table 3-4. DEM errors description

Function	Error Code	Condition
Fr_ControllerInit	FR_E_CTRL_TESTRESULT	CC is not accessible CC can not be disabled CC can not enter POC:Config state

Table continues on the next page...

Table 3-4. DEM errors description (continued)

Function	Error Code	Condition
		There are errors when driver configures the transmit, receive message buffers, shadow buffers and FIFO arrays CC can not leave POC: Config state There are errors when reading back or incorrect value when comparing cluster and configuration values to reference values held in the configuration Some interrupt flag was not clear or some interrupt was not disabled
Fr_StartCommunication	FR_E_CTRL_TESTRESULT	CC is not accessible CHI command was not accepted by PE due to BSY flag
Fr_AllowColdstart	FR_E_CTRL_TESTRESULT	CC is not accessible CHI command was not accepted by PE due to BSY flag
Fr_AllowColdstart	FR_E_CTRL_TESTRESULT	CC is not accessible CHI command was not accepted by PE due to BSY flag
Fr_HaltCommunication	FR_E_CTRL_TESTRESULT	CC is not accessible CHI command was not accepted by PE due to BSY flag
Fr_AbortCommunication	FR_E_CTRL_TESTRESULT	CC is not accessible CHI command was not accepted by PE due to BSY flag
Fr_SendWUP	FR_E_CTRL_TESTRESULT	CC is not accessible CHI command was not accepted by PE due to BSY flag
Fr_SetWakeupChannel	FR_E_CTRL_TESTRESULT	CC is not accessible Error occurred during set wakeup channel
Fr_GetPOCStatus	FR_E_CTRL_TESTRESULT	CC is not accessible Can get POC status due to hardware errors
Fr_GetGlobalTime	FR_E_CTRL_TESTRESULT	CC is not accessible Incorrect cycle and macrotick value reading from registers
Fr_SetAbsoluteTimer	FR_E_CTRL_TESTRESULT	CC is not accessible
Fr_CancelAbsoluteTimer	FR_E_CTRL_TESTRESULT	CC is not accessible
Fr_EnableAbsoluteTimerIRQ	FR_E_CTRL_TESTRESULT	CC is not accessible
Fr_AckAbsoluteTimerIRQ	FR_E_CTRL_TESTRESULT	CC is not accessible
Fr_DisableAbsoluteTimerIRQ	FR_E_CTRL_TESTRESULT	CC is not accessible
Fr_GetAbsoluteTimerIRQStatus	FR_E_CTRL_TESTRESULT	CC is not accessible
Fr_GetNmVector	FR_E_CTRL_TESTRESULT	CC is not accessible Hardware errors may occur
Fr_GetChannelStatus	FR_E_CTRL_TESTRESULT	CC is not accessible
Fr_GetClockCorrection	FR_E_CTRL_TESTRESULT	CC is not accessible Incorrect rate and offset correction values from FR_RTCORVR and FR_OFCORVR registers
Fr_GetSyncFrameList	FR_E_CTRL_TESTRESULT	CC is not accessible

Table continues on the next page...

Table 3-4. DEM errors description (continued)

Function	Error Code	Condition
		Hardware errors may occur
Fr_GetWakeupRxStatus	FR_E_CTRL_TESTRESULT	CC is not accessible Can not clear flag in register FR_PSR3
Fr_TransmitTxLPdu	FR_E_CTRL_TESTRESULT	CC is not accessible MB is not configured for Tx The MB is not locked
Fr_ReceiveRxLPdu	FR_E_CTRL_TESTRESULT	CC is not accessible MB is not configured for Rx
Fr_ReceiveRxLPdu	FR_E_LPDU_SLOTSTATUS	Single slot status error bit is set
Fr_CheckTxLPduStatus	FR_E_CTRL_TESTRESULT	CC is not accessible Data was not ready for transmission
Fr_CheckTxLPduStatus	FR_E_LPDU_SLOTSTATUS	CC is not accessible Single slot status error bit is set
Fr_PrepareLPdu	FR_E_CTRL_TESTRESULT	CC is not accessible Hardware error, can not enable, disable Message Buffer or invalid MB configuration
Fr_CancelTxLPdu	FR_E_CTRL_TESTRESULT	CC is not accessible Data not ready for transmission Can not lock or unlock the MB
Fr_ReconfigLPdu	FR_E_CTRL_TESTRESULT	CC is not accessible Data not ready for transmission Can not enable or disable MB
Fr_DisableLPdu	FR_E_CTRL_TESTRESULT	CC is not accessible Can not disable the MB
Fr_ReadCCConfig	FR_E_CTRL_TESTRESULT	CC is not accessible Incorrect rate and offset correction values from FR_RTCORVR and FR_OFCORVR registers
Fr_GetNumOfStartupFrames	FR_E_CTRL_TESTRESULT	CC is not accessible
Fr_DeInit	FR_E_CTRL_TESTRESULT	Hardware error or incorrect sequence

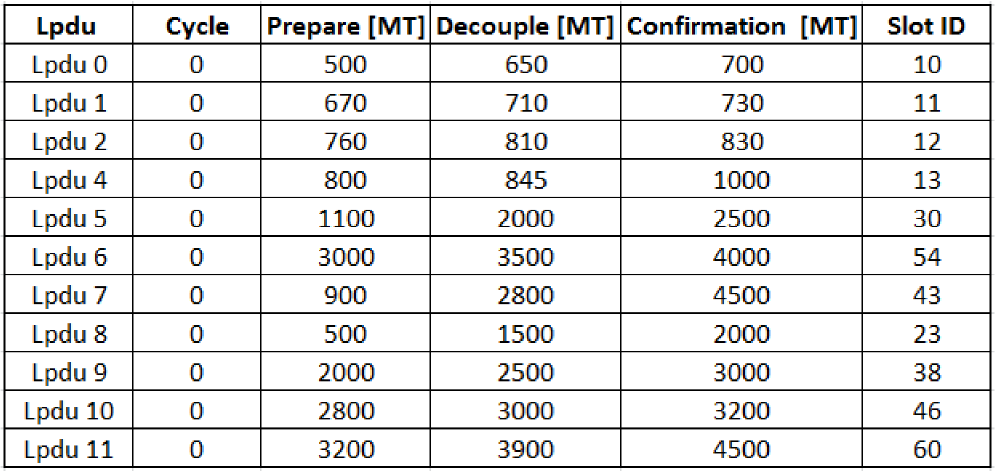
3.4 Function Definitions

API description of all functions supported by the driver can be found in the AUTOSAR FlexRay Driver software specification document [Table 2-3].

3.4.1 Reconfiguration Concept

In case of insufficient number of available message buffers on the FlexRay CC the message buffer reconfiguration can be used.

The FlexRay driver implements the reconfiguration approach in the following way: If the `Force Reconfiguration` and `Prepare LPdu Support` parameters are enabled, the `Fr_PrepareLPdu()` function also is called. Base on the job list and slot id of each lpdu, the driver will figure out the lpdus which dose not overlap the timestamp. The timestamp of each lpdu is calculated base on "starttime" and "endtime" of that lpdu. The "starttime" is prepare state and the "endtime" is transmission confrimation state. Between "starttime" and "endtime" is transmission state. The detail about the states of each lpdu, please see more the "[SWS_FrIf_05134]" requirment. One physical message buffer may be shared for many Lpdus. Following is a example for sharing MB:



Lpdu	Cycle	Prepare [MT]	Decouple [MT]	Confirmation [MT]	Slot ID
Lpdu 0	0	500	650	700	10
Lpdu 1	0	670	710	730	11
Lpdu 2	0	760	810	830	12
Lpdu 4	0	800	845	1000	13
Lpdu 5	0	1100	2000	2500	30
Lpdu 6	0	3000	3500	4000	54
Lpdu 7	0	900	2800	4500	43
Lpdu 8	0	500	1500	2000	23
Lpdu 9	0	2000	2500	3000	38
Lpdu 10	0	2800	3000	3200	46
Lpdu 11	0	3200	3900	4500	60

These lpdus will be organize into 2 group. these are [lpdu0,lpdu2,lpdu5,lpdu10,lpdu11] and [lpdu6,lpdu8].

What happens during the reconfiguration is: The message buffer is disabled, all necessary configuration data are stored and buffer is enabled. Since the driver does not store any information about actual message buffer configuration, the `Fr_PrepareLPdu()` function has to be called before each operation with physical resource configured for reconfiguration (e.g. before `Fr_TransmitTxLPdu()` function).

3.4.2 FIFO Operations

The FlexRay driver implements FIFO buffers in the following way: If the `Fr_ReceiveRxLPdu()` function is called, the driver figures out the physical resource (i.e. either individual message buffer or FIFO storage) mapped to the processing of the FlexRay frame identified by given `Fr_LPduIdx` and uses either an individual receive message buffer or one of FIFO storages for other operations. In case of reception by FIFO storage, an upper software layer (e.g. application or FlexRay Interface) has to ensure that the `Fr_ReceiveRxLPdu()` function is called so frequently to serve all received frames. Driver does not implement mechanism of processing a FIFO overrun error, which may occur due to full FIFO storage. Since there is no way how to notify application about this error flag user is responsible for FIFO servicing.

Note

FIFO buffers can not be used for reconfiguration

3.5 API Reference

This section contains description of the FR driver API which is implemented as defined in the AUTOSAR FlexRay Driver software specification document [Table 2-3].

3.5.1 Function Fr_Init

Controller initialization function.

Prototype: `void Fr_Init(const Fr_ConfigType *Fr_ConfigPtr);`

Table 3-5. Fr_Init Arguments

Type	Name	Direction	Description
const `Fr_ConfigType` *	Fr_ConfigPtr	**input**	Pointer to the Controller configuration

Return: none

[SWS_Fr_00137]: This API internally stores the configuration data address to enable subsequent API calls to access the configuration data, activates FlexRay CC and immediately set them into POC:Halt state. If development error detection is enabled the successful initialization shall be remembered internally for other API functions to check for proper module initialization. If this API function detects errors while accessing any CC (for example a protocol control command has not been accepted), it calls Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return

Note

On the Mamba platform, the enable/disable FR module is can not cleared after it is set. This is a limitation of hardware. Therefore, the Flexray module is intialized only one time in all runtime. If Flexray module is intialized two times, some Flexray registers can not updated. That is a error and one error notification "FrErrorInitiNotification" is created for this. If meet this error, user should be to stop all operate, restart microcontroller anh check again application in order to make sure that Flexray module is intialized one time.

3.5.2 Function Fr_ControllerInit

CC configuration.

Prototype: `Std_ReturnType Fr_ControllerInit(uint8 Fr_CtrlIdx);`

Table 3-6. Fr_ControllerInit Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver

Return: Std_ReturnType

- E_OK API API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00148]: Switch CC into POC:Config, configures all FlexRay cluster and node configuration parameters, configures all transmit/receive resources and switch CC into POC:Ready

3.5.3 Function Fr_DeInit

Controller De-initialization function.

Prototype: `void Fr_DeInit(void);`

Table 3-7. Fr_DeInit Arguments

Type	Name	Direction	Description
void	none	**input**	This function has the input parameter as void type.

Return: none

[CPR-MCAL-825.fr]: The MCAL drivers shall be able to run in a non-privileged processor mode (e.g. User mode). All known related constraints shall be documented. A vendor specific pre-compile boolean configuration parameter (Mdl)EnableUserModeSupport {(Mdl)_ENABLE_YSER_MODE_SUPPORT}shall be created for each driver to activate the specific implementation for non-privileged mode. By default, '(Mdl)EnableUserModeSupport' field shall be disabled.

Note

On other platforms which are support IpVault version lower version 10, the enable/disable FR module bit is can not cleared after it is set. This is a limitation of hardware. Therefore, the Fr_DeInit function will be rejected on these platforms.

3.5.4 Function Fr_StartCommunication

Starts the communication.

Prototype: `Std_ReturnType Fr_StartCommunication(uint8 Fr_CtrlIdx);`

Table 3-8. Fr_StartCommunication Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver

Return: Std_ReturnType
- E_OK API call has been successful
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00177]: Invoke the CC CHI command RUN, which initiates the startup procedure within the FlexRay CC

3.5.5 Function Fr_AllowColdstart

Invokes the CC CHI command ALLOW_COLDSTART.

Prototype: `Std_ReturnType Fr_AllowColdstart(uint8 Fr_CtrlIdx);`

Table 3-9. Fr_AllowColdstart Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver

Return: Std_ReturnType
- E_OK API call has been successful
- E_NOT_OK API call in bad POC state or failure in access to the controller

[SWS_Fr_00182]: Invokes the CC CHI command ALLOW_COLDSTART

3.5.6 Function Fr_AllSlots

Invokes the CC CHI command ALL_SLOTS.

Prototype: `Std_ReturnType Fr_AllSlots(uint8 Fr_CtrlIdx);`

Table 3-10. Fr_AllSlots Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver

Return: Std_ReturnType

- E_OK API call has been successful
- E_NOT_OK API call in bad POC state or failure in access to the controller

[SWS_Fr_00518]: Invoke the CC CHI command ALL_SLOTS, which requests a switch from key slot only mode to all slots transmission mode at the beginning of the next communication cycle.

3.5.7 Function Fr_HaltCommunication

This API call stops communication.

Prototype: `Std_ReturnType Fr_HaltCommunication(uint8 Fr_CtrlIdx);`

Table 3-11. Fr_HaltCommunication Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver

Return: Std_ReturnType

- E_OK API call has been successful
- E_NOT_OK API call in bad POC state or failure in access to the controller

[SWS_Fr_00187]: Invoke the CC CHI command HALT, which requests to halt the communication at the end of the current FlexRay communication cycle

3.5.8 Function Fr_AbortCommunication

Abort the communication.

Prototype: `Std_ReturnType Fr_AbortCommunication(uint8 Fr_CtrlIdx);`

Table 3-12. Fr_AbortCommunication Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver

Return: Std_ReturnType

- E_OK API call has been successful
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00191]: Invoke the CHI command FREEZE, which immediately aborts the communication and force the CC to the POC:Halt

Note

On the Mamba platform have one errata "e2302" related the Feeze mode. Follow the solution of errata, the FEEZE command will be replaced bay HALT command.

3.5.9 Function Fr_SendWUP

Initiates transition to POC:Wakeup.

Prototype: `Std_ReturnType Fr_SendWUP(uint8 Fr_CtrlIdx);`

Table 3-13. Fr_SendWUP Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00196]: Invoke the CC CHI command WAKEUP, which initiates the Wakeup Symbol transmission procedure on the configured FlexRay channel

3.5.10 Function Fr_SetWakeupChannel

Selects which channel sends WUPs.

Prototype: `Std_ReturnType Fr_SetWakeupChannel(uint8 Fr_CtrlIdx, Fr_ChannelType Fr_ChnlIdx);`

Table 3-14. Fr_SetWakeupChannel Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
`Fr_ChannelType`	Fr_ChnlIdx	**input**	Index of FlexRay channel within the context of the FlexRay CC Fr_CtrlIdx

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00202]: Change the CCs POCState to POC:Config by invoking the CHI command CONFIG, configure the wakeup channel according to parameter Fr_ChnlIdx and change the CCs POCState to POC:Ready again by invoking the CHI command CONFIG_COMPLETE

3.5.11 Function Fr_GetPOCStatus

Query for the controller status.

Prototype: `Std_ReturnType Fr_GetPOCStatus(uint8 Fr_CtrlIdx, Fr_POCStatusType *Fr_POCStatusPtr);`

Table 3-15. Fr_GetPOCStatus Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
`Fr_POCStatusType` *	Fr_POCStatusPtr	**output**	Address the output value is stored to

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00217]: Query the CCs actual POC status by reading the CHI variable vPOC and write the result to parameter Fr_POCStatusPtr.

3.5.12 Function Fr_TransmitTxLPdu

Update selected message buffer with new data.

Prototype: `Std_ReturnType Fr_TransmitTxLPdu(uint8 Fr_CtrlIdx, uint16 Fr_LPduIdx, const uint8 *Fr_LSduPtr, uint8 Fr_LSduLength);`

Table 3-16. Fr_TransmitTxLPdu Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
uint16	Fr_LPduIdx	**input**	The index is used to uniquely identify a FlexRay frame
const uint8 *	Fr_LSduPtr	**input**	Pointer to a buffer where the assembled LSdu to be transmitted within this LPdu is stored at
uint8	Fr_LSduLength	**input**	Determines the length of the data (in Bytes) to be transmitted

Return: Std_ReturnType

- E_OK API call has been successful
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00224]: Figures out the physical resource mapped to the transmission of the FlexRay frame identified by Fr_LPduIdx, copies Fr_LSduLength bytes from address Fr_LSduPtr into the FlexRay CCs transmission resource and activate it for transmission

3.5.13 Function Fr_CancelTxLPdu

Cancels the already pending transmission of a message buffer.

Prototype: `Std_ReturnType Fr_CancelTxLPdu(uint8 Fr_CtrlIdx, uint16 Fr_LPduIdx);`

Table 3-17. Fr_CancelTxLPdu Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
uint16	Fr_LPduIdx	**input**	This index is used to uniquely identify a FlexRay frame

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00611]: Figure out the physical resource (e.g., a buffer) mapped to the transmission of the FlexRay frame identified by Fr_LpduIdx. If the physical resource figured out is actively pending for transmission, then the transmit request of this particular resource shall be terminated and E_OK returned. If no transmission is pending E_NOT_OK shall be returned, indicating that no such cancelation took place.

3.5.14 Function Fr_ReceiveRxLPdu

Receives data.

Prototype: `Std_ReturnType Fr_ReceiveRxLPdu(uint8 Fr_CtrlIdx, uint16 Fr_LPduIdx, uint8 *Fr_LSduPtr, Fr_RxLPduStatusType *Fr_LPduStatusPtr, uint8 *Fr_LSduLengthPtr);`

Table 3-18. Fr_ReceiveRxLPdu Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
uint16	Fr_LPduIdx	**input**	This index is used to uniquely identify a FlexRay frame
uint8 *	Fr_LSduPtr	**output**	Pointer to a buffer where the LSdu
`Fr_RxLPduStatusType` *	Fr_LPduStatusPtr	**output**	Pointer to the memory location where the status of the LPdu shall be stored
uint8 *	Fr_LSduLengthPtr	**output**	Pointer to the memory location where the length of the LSdu (in bytes) shall be stored

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00233]: This function figures out the physical resource mapped to the reception of the FlexRay frame identified by Fr_LPduIdx, figures out whether a new FlexRay frame has been received. If a new frame has been received, copies the received payload data to address Fr_LSduPtr, stores the number of bytes and the status `FR_RECEIVED`. If a new frame has been received, FIFO code is enabled and Fr_LPduIdx is associated with FIFO A or FIFO B storage, the received payload data are copied to address Fr_LSduPtr, the number of bytes is stored and the status `FR_RECEIVED` is returned. If no new frame has been received, the function doesnt copy any payload data to Fr_LSduPtr, writes 0 to parameter Fr_LSduLengthPtr and stores the status `FR_NOT_RECEIVED`.

3.5.15 Function Fr_CheckTxLPduStatus

Checks if data have been transmitted.

Prototype: `Std_ReturnType Fr_CheckTxLPduStatus(uint8 Fr_CtrlIdx, uint16 Fr_LPduIdx, Fr_TxLPduStatusType *Fr_TxLPduStatusPtr);`

Table 3-19. Fr_CheckTxLPduStatus Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
uint16	Fr_LPduIdx	**input**	This index is used to uniquely identify a FlexRay frame

Table continues on the next page...

Table 3-19. Fr_CheckTxLPduStatus Arguments (continued)

Type	Name	Direction	Description
`Fr_TxLPduStatusType *`	Fr_TxLPduStatusPtr	**output**	Pointer used to store the transmit status of the LSdu

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00244]: Figures out the physical resource mapped to the transmission of the FlexRay frame identified by Fr_LPduIdx and check whether the transmission resource is still pending for transmission or not

3.5.16 Function Fr_PrepareLPdu

Reconfigures physical resource.

Prototype: `Std_ReturnType Fr_PrepareLPdu(uint8 Fr_CtrlIdx, uint16 Fr_LPduIdx);`

Table 3-20. Fr_PrepareLPdu Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
uint16	Fr_LPduIdx	**input**	This index is used to uniquely identify a FlexRay frame

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00249]: Figures out the physical resource mapped to the processing of the FlexRay frame identified by Fr_LPduIdx and configure the physical resource appropriate for LPduIdx operation if required by Fr configuration

CAUTION

Only receive buffers and single transmit buffers reconfiguration is supported (not FIFO)

3.5.17 Function Fr_ReconfigLPdu

Reconfigures a given LPdu according to the input parameters.

Prototype: `Std_ReturnType Fr_ReconfigLPdu(uint8 Fr_CtrlIdx, uint16 Fr_LPduIdx, uint16 Fr_FrameId, Fr_ChannelType Fr_ChnlIdx, uint8 Fr_CycleRepetition, uint8 Fr_CycleOffset, uint8 Fr_PayloadLength, uint16 Fr_HeaderCRC);`

Table 3-21. Fr_ReconfigLPdu Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
uint16	Fr_LPduIdx	**input**	This index is used to uniquely identify a FlexRay frame
uint16	Fr_FrameId	**input**	FlexRay Frame ID the FrIf_LPdu shall be configured to
`Fr_ChannelType`	Fr_ChnlIdx	**input**	FlexRay Channel the FrIf_LPdu shall be configured to
uint8	Fr_CycleRepetition	**input**	Cycle Repetition part of the cycle filter mechanism FrIf_LPdu shall be configured to
uint8	Fr_CycleOffset	**input**	Cycle Offset part of the cycle filter mechanism FrIf_LPdu shall be configured to
uint8	Fr_PayloadLength	**input**	Payloadlength in units of bytes the FrIf_LPduIdx shall be configured to
uint8	Fr_HeaderCRC	**input**	Header CRC the FrIf_LPdu shall be configured to

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00525]: Figure out the physical resource (e.g., a buffer) mapped to the processing of the FlexRay frame as identified by Fr_LpduIdx. Configure the physical resource (a buffer) according to the parameters given at the API. The Lpdu direction is statically associated with the Lpdu and cannot be changed by this service.

3.5.18 Function Fr_DisableLPdu

Disables the hardware resource of a LPdu for transmission/reception.

Prototype: `Std_ReturnType Fr_DisableLPdu(uint8 Fr_CtrlIdx, uint16 Fr_LPduIdx);`

Table 3-22. Fr_DisableLPdu Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
uint8	Fr_LPduIdx	**input**	This index is used to uniquely identify a FlexRay frame

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00540]: Figure out the physical resource (e.g., a buffer) mapped to the processing of the FlexRay frame identified by Fr_LpduIdx. Configure the physical resource (a buffer) in a way that it does not take part in the transmission/reception process.

3.5.19 Function Fr_GetGlobalTime

Function gets FlexRay cluster global time.

Prototype: `Std_ReturnType Fr_GetGlobalTime(uint8 Fr_CtrlIdx, uint8 *Fr_CyclePtr, uint16 *Fr_MacroTickPtr);`

Table 3-23. Fr_GetGlobalTime Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
uint8 *	Fr_CyclePtr	**output**	Address where the current FlexRay communication cycle value shall be stored
uint16 *	Fr_MacroTickPtr	**output**	Address where the current macrotick value shall be stored

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00256]: Read the current global FlexRay time and writes it to the output parameters Fr_cyclePtr and Fr_MacrotickPtr

3.5.20 Function Fr_GetNmVector

Read network management vector.

Prototype: `Std_ReturnType Fr_GetNmVector(uint8 Fr_CtrlIdx, uint8 *Fr_NmVectorPtr);`

Table 3-24. Fr_GetNmVector Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
uint8 *	Fr_NmVectorPtr	**output**	Address where the NmVector of the last communication cycle shall be stored

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00262]: Read the network management vector of the last communication cycle and write it to the output parameter Fr_NmVectorPtr. The number of bytes written to the output parameter is constant and known at configuration time

3.5.21 Function Fr_GetNumOfStartupFrames

Gets the current number of startup frames seen on the cluster.

Prototype: `Std_ReturnType Fr_GetNumOfStartupFrames(uint8 Fr_CtrlIdx, uint8* Fr_NumOfStartupFramesPtr);`

Table 3-25. Fr_GetNumOfStartupFrames Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
uint8*	Fr_NumOfStartupFramesPtr	**output**	Address where the number of startup frames seen within the last even/odd cycle pair shall be stored

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00549]: Read the number of aligned startup frame pairs received or transmitted during the previous double cycle, aggregated across both channels and write it to the output parameter Fr_NumOfStartupFramesPtr.

3.5.22 Function Fr_GetChannelStatus

Gets the channel status information.

Prototype: `Std_ReturnType Fr_GetChannelStatus(uint8 Fr_CtrlIdx, uint16* Fr_ChannelAStatusPtr, uint16* Fr_ChannelBStatusPtr);`

Table 3-26. Fr_GetChannelStatus Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
uint16*	Fr_ChannelAStatusPtr	**output**	Address where the bitcoded channel A status information shall be stored
uint16*	Fr_ChannelBStatusPtr	**output**	Address where the bitcoded channel B status information shall be stored

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00558]: Read the aggregated channel status, NIT status, symbol window status and write it to the output parameter Fr_ChannelAStatusPtr/Fr_ChannelBStatusPtr.

3.5.23 Function Fr_GetClockCorrection

Gets the current clock correction values.

Prototype: `Std_ReturnType Fr_GetClockCorrection(uint8 Fr_CtrlIdx, sint16* Fr_RateCorrectionPtr, sint32* Fr_OffsetCorrectionPtr);`

Table 3-27. Fr_GetClockCorrection Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
sint16*	Fr_RateCorrectionPtr	**output**	Address where the current rate correction value shall be stored
sint32*	Fr_OffsetCorrectionPtr	**output**	Address where the current offset correction value shall be stored

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00566]: Read the rate correction value (vInterimRateCorrection) and write it as signed integer to the output parameter Fr_RateCorrectionPtr. Read the offset correction value (vInterimOffsetCorrection) and write it as signed integer to the output parameter Fr_OffsetCorrectionPtr.

3.5.24 Function Fr_GetSyncFrameList

Gets a list of syncframes received or transmitted on channel A and channel B via the even and odd communication cycle.

Prototype: `Std_ReturnType Fr_GetSyncFrameList(uint8 Fr_CtrlIdx, uint8 Fr_ListSize, uint16* Fr_ChannelAEvenListPtr, uint16* Fr_ChannelBEvenListPtr, uint16* Fr_ChannelAOddListPtr, uint16* Fr_ChannelBOddListPtr);`

Table 3-28. Fr_GetSyncFrameList Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
uint8	Fr_ListSize	**input**	Size of the arrays passed via parameters: Fr_ChannelAEvenListPtr Fr_ChannelBEvenListPtr Fr_ChannelAOddListPtr Fr_ChannelBOddListPtr
uint16*	Fr_ChannelAEvenListPtr	**output**	Address the list of syncframes on channel A within the even communication cycle is written to
uint16*	Fr_ChannelBEvenListPtr	**output**	Address the list of syncframes on channel B within the even communication cycle is written to
uint16*	Fr_ChannelAOddListPtr	**output**	Address the list of syncframes on channel A within the odd communication cycle is written to
uint16*	Fr_ChannelBOddListPtr	**output**	Address the list of syncframes on channel B within the odd communication cycle is written to

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors or Fr_ListSize=0

[SWS_Fr_00575]: Read the list of syncframes received in the last even communication cycle on channel A and write it as array to the memory location Fr_ChannelAEvenListPtr. Read the list of syncframes received in the last even communication cycle on channel B and write it as array to the memory location Fr_ChannelBEvenListPtr. Read the list of syncframes received in the last odd communication cycle on channel A and write it as array to the memory location Fr_ChannelAOddListPtr. Read the list of syncframes received in the last odd communication cycle on channel B and write it as array to the memory location Fr_ChannelBOddListPtr.

3.5.25 Function Fr_GetWakeupRxStatus

Gets the wakeup received information from the PSR3 register.

Prototype: `Std_ReturnType Fr_GetWakeupRxStatus(uint8 Fr_CtrlIdx, uint8* Fr_WakeupRxStatusPtr);`

Table 3-29. Fr_GetWakeupRxStatus Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
uint8*	Fr_WakeupRxStatusPtr	**output**	Address where bitcoded wakeup reception status shall be stored

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00588]: Read the wakeup pattern received indicators for channel A and channel B and write it to the output parameter Fr_WakeupRxStatusPtr.

3.5.26 Function Fr_SetAbsoluteTimer

Sets the absolute FlexRay timer.

Prototype: `Std_ReturnType Fr_SetAbsoluteTimer(uint8 Fr_CtrlIdx, uint8 Fr_AbsTimerIdx, uint8 Fr_Cycle, uint16 Fr_Offset);`

Table 3-30. Fr_SetAbsoluteTimer Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
uint8	Fr_AbsTimerIdx	**input**	Index of absolute timer within the context of the FlexRay CC
uint8	Fr_Cycle	**input**	Absolute cycle the timer shall elapse in
uint16	Fr_Offset	**input**	Offset within cycle Fr_Cycle in units of macrotick the timer shall elapse at

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00273]: Program the absolute FlexRay timer Fr_AbsTimerIdx according to the parameters Fr_Cycle and Fr_Offset

3.5.27 Function Fr_CancelAbsoluteTimer

Stops an absolute timer.

Prototype: `Std_ReturnType Fr_CancelAbsoluteTimer(uint8 Fr_CtrlIdx, uint8 Fr_AbsTimerIdx);`

Table 3-31. Fr_CancelAbsoluteTimer Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
uint8	Fr_AbsTimerIdx	**input**	Index of absolute timer within the context of the FlexRay CC

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00287]: Stop the absolute timer Fr_AbsTimerIdx

3.5.28 Function Fr_EnableAbsoluteTimerIRQ

Enables absolute timer interrupts.

Prototype: `Std_ReturnType Fr_EnableAbsoluteTimerIRQ(uint8 Fr_CtrlIdx, uint8 Fr_AbsTimerIdx);`

Table 3-32. Fr_EnableAbsoluteTimerIRQ Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
uint8	Fr_AbsTimerIdx	**input**	Index of absolute timer within the context of the FlexRay CC

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00298]: Enable the interrupt line related to timer Fr_AbsTimerIdx

3.5.29 Function Fr_AckAbsoluteTimerIRQ

Clears absolute timer interrupt flag.

Prototype: `Std_ReturnType Fr_AckAbsoluteTimerIRQ(uint8 Fr_CtrlIdx, uint8 Fr_AbsTimerIdx);`

Table 3-33. Fr_AckAbsoluteTimerIRQ Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
uint8	Fr_AbsTimerIdx	**input**	Index of absolute timer within the context of the FlexRay CC

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00309]: Reset the interrupt condition of absolute timer Fr_AbsTimerIdx

3.5.30 Function Fr_DisableAbsoluteTimerIRQ

Disables absolute timer interrupt generation.

Prototype: `Std_ReturnType Fr_DisableAbsoluteTimerIRQ(uint8 Fr_CtrlIdx, uint8 Fr_AbsTimerIdx);`

Table 3-34. Fr_DisableAbsoluteTimerIRQ Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
uint8	Fr_AbsTimerIdx	**input**	Index of absolute timer within the context of the FlexRay CC

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00320]: Disable the interrupt line related to absolute timer Fr_AbsTimerIdx

3.5.31 Function Fr_GetAbsoluteTimerIRQStatus

Checks if the absolute timer flag is set.

Prototype: `Std_ReturnType Fr_GetAbsoluteTimerIRQStatus(uint8 Fr_CtrlIdx, uint8 Fr_AbsTimerIdx, boolean *Fr_IRQStatusPtr);`

Table 3-35. Fr_GetAbsoluteTimerIRQStatus Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
uint8	Fr_AbsTimerIdx	**input**	Index of absolute timer within the context of the FlexRay CC
boolean *	Fr_IRQStatusPtr	**output**	Address the output value is stored to

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

[SWS_Fr_00332]: Check whether the interrupt of absolute timer Fr_AbsTimerIdx is pending. Writes (VAR(boolean, FR_VAR))TRUE to output parameter Fr_IRQStatusPtr in case the interrupt is pending, (VAR(boolean, FR_VAR))FALSE otherwise

3.5.32 Function Fr_GetVersionInfo

Software module version query.

Prototype: `void Fr_GetVersionInfo(Std_VersionInfoType *VersioninfoPtr);`

Table 3-36. Fr_GetVersionInfo Arguments

Type	Name	Direction	Description
Std_VersionInfoType *	VersioninfoPtr	**output**	Std_VersionInfoType type pointer where to store version numbers

Return: none

[SWS_Fr_00341]: The function Fr_GetVersionInfo shall return the version information of this module

3.5.33 Function Fr_ReadCCConfig

Reads a FlexRay protocol configuration parameter from PCR register.

Prototype: `Std_ReturnType Fr_ReadCCConfig(uint8 Fr_CtrlIdx, uint8 Fr_ConfigParamIdx, uint32* Fr_ConfigParamValuePtr);`

Table 3-37. Fr_ReadCCConfig Arguments

Type	Name	Direction	Description
uint8	Fr_CtrlIdx	**input**	Index of FlexRay CC within the context of the FlexRay Driver
uint8	Fr_ConfigParamIdx	**input**	Index that identifies the configuration parameter to read. See macros FR_CIDX_<config_parameter_name>
uint32*	Fr_ConfigParamValuePtr	**output**	Address the output value is stored to

Return: Std_ReturnType

- E_OK API call finished successfully
- E_NOT_OK API call aborted due to errors

Read the value of the configuration parameter requested by Fr_ConfigParamIdx from the configuration and write it to output parameter *Fr_ConfigParamValuePtr.

3.5.34 Function Index

Table 3-38. Quick Function Reference

Type	Name	Arguments
Std_ReturnType	**Fr_AbortCommunication**	uint8 Fr_CtrlIdx
Std_ReturnType	**Fr_AckAbsoluteTimerIRQ**	uint8 Fr_CtrlIdx uint8 Fr_AbsTimerIdx
Std_ReturnType	**Fr_AllowColdstart**	uint8 Fr_CtrlIdx
Std_ReturnType	**Fr_AllSlots**	uint8 Fr_CtrlIdx
Std_ReturnType	**Fr_CancelAbsoluteTimer**	uint8 Fr_CtrlIdx uint8 Fr_AbsTimerIdx
Std_ReturnType	**Fr_CheckTxLPduStatus**	uint8 Fr_CtrlIdx uint16 Fr_LPduIdx `Fr_TxLPduStatusType` * Fr_TxLPduStatusPtr
Std_ReturnType	**Fr_ControllerInit**	uint8 Fr_CtrlIdx
Std_ReturnType	**Fr_DisableAbsoluteTimerIRQ**	uint8 Fr_CtrlIdx uint8 Fr_AbsTimerIdx
Std_ReturnType	**Fr_DisableLPdu**	uint8 Fr_CtrlIdx uint16 Fr_LPduIdx
Std_ReturnType	**Fr_EnableAbsoluteTimerIRQ**	uint8 Fr_CtrlIdx uint8 Fr_AbsTimerIdx
Std_ReturnType	**Fr_GetAbsoluteTimerIRQStatus**	uint8 Fr_CtrlIdx uint8 Fr_AbsTimerIdx boolean * Fr_IRQStatusPtr
Std_ReturnType	**Fr_GetChannelStatus**	uint8 Fr_CtrlIdx uint16* Fr_ChannelAStatusPtr uint16* Fr_ChannelBStatusPtr
Std_ReturnType	**Fr_GetClockCorrection**	uint8 Fr_CtrlIdx sint16* Fr_RateCorrectionPtr sint32* Fr_OffsetCorrectionPtr
Std_ReturnType	**Fr_GetGlobalTime**	uint8 Fr_CtrlIdx uint8 * Fr_CyclePtr uint16 * Fr_MacroTickPtr
Std_ReturnType	**Fr_GetNmVector**	uint8 Fr_CtrlIdx uint8 * Fr_NmVectorPtr
Std_ReturnType	**Fr_GetNumOfStartupFrames**	uint8 Fr_CtrlIdx uint8* Fr_NumOfStartupFramesPtr
Std_ReturnType	**Fr_GetPOCStatus**	uint8 Fr_CtrlIdx `Fr_POCStatusType` * Fr_POCStatusPtr
Std_ReturnType	**Fr_GetSyncFrameList**	uint8 Fr_CtrlIdx uint8 Fr_ListSize uint16* Fr_ChannelAEvenListPtr uint16* Fr_ChannelBEvenListPtr uint16* Fr_ChannelAOddListPtr uint16* Fr_ChannelBOddListPtr
void	**Fr_GetVersionInfo**	Std_VersionInfoType * VersioninfoPtr
Std_ReturnType	**Fr_GetWakeupRxStatus**	uint8 Fr_CtrlIdx uint8* Fr_WakeupRxStatusPtr
Std_ReturnType	**Fr_HaltCommunication**	uint8 Fr_CtrlIdx

Table continues on the next page...

Table 3-38. Quick Function Reference (continued)

Type	Name	Arguments
void	**Fr_Init**	const Fr_ConfigType * Fr_ConfigPtr
Std_ReturnType	**Fr_PrepareLPdu**	uint8 Fr_CtrlIdx uint16 Fr_LPduIdx
Std_ReturnType	**Fr_ReadCCConfig**	uint8 Fr_CtrlIdx uint8 Fr_ConfigParamIdx uint32* Fr_ConfigParamValuePtr
Std_ReturnType	**Fr_ReceiveRxLPdu**	uint8 Fr_CtrlIdx uint16 Fr_LPduIdx uint8 * Fr_LSduPtr Fr_RxLPduStatusType * Fr_LPduStatusPtr uint8 * Fr_LSduLengthPtr
Std_ReturnType	**Fr_ReconfigLPdu**	uint8 Fr_CtrlIdx uint16 Fr_LPduIdx uint16 Fr_FrameId Fr_ChannelType Fr_ChnlIdx uint8 Fr_CycleRepetition uint8 Fr_CycleOffset uint8 Fr_PayloadLength uint16 Fr_HeaderCRC
Std_ReturnType	**Fr_SendWUP**	uint8 Fr_CtrlIdx
Std_ReturnType	**Fr_SetAbsoluteTimer**	uint8 Fr_CtrlIdx uint8 Fr_AbsTimerIdx uint8 Fr_Cycle uint16 Fr_Offset
Std_ReturnType	**Fr_SetWakeupChannel**	uint8 Fr_CtrlIdx Fr_ChannelType Fr_ChnlIdx
Std_ReturnType	**Fr_StartCommunication**	uint8 Fr_CtrlIdx
Std_ReturnType	**Fr_TransmitTxLPdu**	uint8 Fr_CtrlIdx uint16 Fr_LPduIdx const uint8 * Fr_LSduPtr uint8 Fr_LSduLength

3.6 Configuration Parameters

This FR driver implementation supports pre-compile and post-build time configuration variants. The pre-compile parameters are stored in the Fr_Cfg.h file and in the Fr_Cfg.c file if precompile configuration variant is used. The post-build parameters are stored in the Fr_PBCfg.c file. All configuration files are generated by Tresos configuration tool.

3.6.1 None-Variant aware parameters

None-Variant aware paremeters, their possible values and their meanings are described in the following text. None-Variant aware paremeters are implemented as preprocessor defines and generated in Fr_Cfg.h

Table 3-39. FrDevErrorDetection

Description	Defines whether DET errors reporting should be included at compile time (`STD_ON`) or excluded (`STD_OFF`).
Class	Autosar Parameter
Range	True, False
Default	True
Source File	Fr_Cfg.h
Source Representation	`#define FR_DEV_ERROR_DETECT <STD_OFF, STD_ON>`

Table 3-40. FrVersionInfoApi

Description	Defines whether version information reporting should be included at compile time (`STD_ON`) or excluded (`STD_OFF`).
Class	Autosar Parameter
Range	True, False
Default	False
Source File	Fr_Cfg.h
Source Representation	`#define FR_VERSION_INFO_API <STD_OFF, STD_ON>`

Table 3-41. FrPrepareLPduSupport

Description	Enables or disables API function Fr_PrepareLPdu (`STD_ON`) or excluded (`STD_OFF`).
Class	Implementation Specific Parameter
Range	True, False
Default	False
Source File	Fr_Cfg.h
Source Representation	`#define FR_PREPARE_LPDU_SUPPORT <STD_OFF, STD_ON>`

Table 3-42. FrReconfigLPduSupport

Description	Enables or disabled API function Fr_ReconfigLPdu (`STD_ON`) or excluded (`STD_OFF`).
Class	Implementation Specific Parameter
Range	True, False

Table continues on the next page...

Table 3-42. FrReconfigLPduSupport (continued)

Default	False
Source File	Fr_Cfg.h
Source Representation	`#define FR_RECONFIG_LPDU_SUPPORT <STD_OFF, STD_ON>`

Table 3-43. FrDisableLPduSupport

Description	Enables or disabled API function Fr_DisableLPdu (`STD_ON`) or excluded (`STD_OFF`).
Class	Implementation Specific Parameter
Range	True, False
Default	False
Source File	Fr_Cfg.h
Source Representation	`#define FR_DISABLE_LPDU_SUPPORT <STD_OFF, STD_ON>`

Table 3-44. FrRxStringentCheck

Description	If stringent check is enabled (`STD_ON`), received frames are only accepted if no slot status error occured.
Class	Implementation Specific Parameter
Range	True, False
Default	False
Source File	Fr_Cfg.h
Source Representation	`#define FR_RXSTRINGENTCHECK <STD_OFF, STD_ON>`

Table 3-45. FrRxStringentLengthCheck

Description	If stringent check is enabled (`STD_ON`), received frames are only accepted the received payload length matches the configured payload length.
Class	Implementation Specific Parameter
Range	True, False
Default	False
Source File	Fr_Cfg.h
Source Representation	`#define FR_RXSTRINGENTLENGTHCHECK <STD_OFF, STD_ON>`

Table 3-46. FrUnusedBitValue

Description	Specifies that the function `Fr_TransmitTxLPdu()` sets the remaining bits in the CC's transmission resource to the configured value given by this parameter.

Table continues on the next page...

Table 3-46. FrUnusedBitValue (continued)

Class	Implementation Specific Parameter
Range	0, 1
Default	0
Source File	Fr_Cfg.h

Source Representation	`/* FrUnusedBitValue parameter value in 8bit format */`
	`#define FR_UNUSED_BIT_VALUE <0x00, 0xFF>U`
	
	`/* FrUnusedBitValue parameter value in 16bit format */`
	`#define FR_UNUSED_BIT_VALUE_16 <0x0000, 0xFFFF>U`
	
	`/* FrUnusedBitValue parameter value in 32bit format. */`
	`#define FR_UNUSED_BIT_VALUE_32 <0x00000000, 0xFFFFFFFF>U`

Table 3-47. Config Variant

Description	Describe sets of configuration parameters.
Class	Autosar Parameter
Range	VariantPostBuild, VariantPreCompile
Default	VariantPostBuild
Source File	Fr_Cfg.h
Source Representation	`#define FR_VARIANT_PRECOMPILE    <STD_OFF, STD_ON>`

Table 3-48. FrDisableDemReportErrorStatus

Description	Defines whether DEM errors/events reporting should be excluded at compile time (`STD_ON`) or included (`STD_OFF`).
Class	Implementation Specific Parameter
Range	True, False
Default	False
Source File	Fr_Cfg.h

Source Representation	`#define FR_DISABLE_DEM_REPORT_ERROR_STATUS <STD_OFF,`
	`STD_ON>`

3.6.2 Variant aware parameters

Variant aware parameters, their possible values and their meanings are described in the following text. The variant aware parameters are implemented as constant structures and arrays stored in flash memory of the MCU and are located in the *Fr_PBcfg_[VariantName].c*. The individual elements of the driver configurations are interlinked to form a tree-like structure which is shown in the figure Figure 3-1.

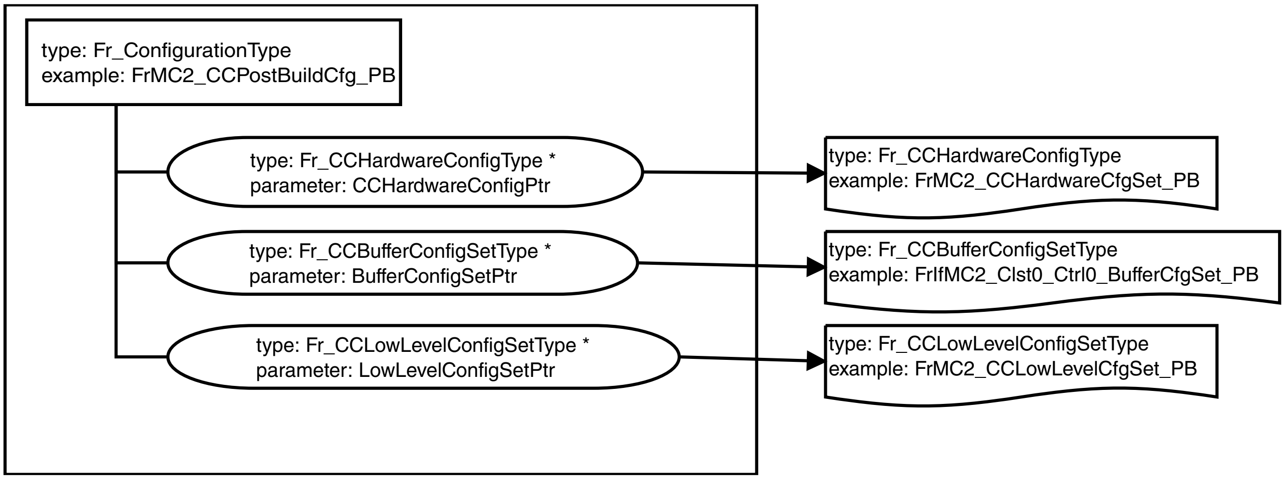

type: Fr_ConfigurationType
example: FrMC2_CCPostBuildCfg_PB
type: Fr_CCHardwareConfigType *
parameter: CCHardwareConfigPtr
type: Fr_CCBufferConfigSetType *
parameter: BufferConfigSetPtr
type: Fr_CCLowLevelConfigSetType *
parameter: LowLevelConfigSetPtr
type: Fr_CCHardwareConfigType
example: FrMC2_CCHardwareCfgSet_PB
type: Fr_CCBufferConfigSetType
example: FrIfMC2_Clst0_Ctrl0_BufferCfgSet_PB
type: Fr_CCLowLevelConfigSetType
example: FrMC2_CCLowLevelCfgSet_PB

Figure 3-1. The Basic Configuration Concept

Each individual configuration structure is described below.

3.6.2.1 Structure Fr_ConfigurationType

The structure of the type `Fr_ConfigurationType` is a top level descriptor containing possible configurations of one FlexRay driver and one multiple configuration. The structure of this type is passed via void pointer to the function `Fr_Init()`.

Declaration

```
typedef struct
{
  CONSTP2CONST(Fr_CCHardwareConfigType, AUTOMATIC, FR_APPL_CONST) CCHardwareConfigPtr;
  CONSTP2CONST(Fr_CCBufferConfigSetType, AUTOMATIC, FR_APPL_CONST) BufferConfigSetPtr;
  CONSTP2CONST(Fr_CCLowLevelConfigSetType, AUTOMATIC, FR_APPL_CONST) LowLevelConfigSetPtr;
  CONSTP2CONST(uint32, AUTOMATIC, FR_APPL_CONST) CCReadBackConfigSetPtr;
  CONSTP2CONST(Fr_DemErrorType, AUTOMATIC, FR_APPL_CONST) FrDemCtrlTestResultPtr;
} Fr_ConfigurationType;
```

Table 3-49. Structure Fr_ConfigurationType member description

Member	Description
CCHardwareConfigPtr	Reference to the hardware configuration structure.
BufferConfigSetPtr	Reference to the buffer configuration structure.

Table continues on the next page...

Table 3-49. Structure Fr_ConfigurationType member description (continued)

Member	Description
LowLevelConfigSetPtr	Reference to the low level configuration structure.
CCReadBackConfigSetPtr	Reference to the array of FlexRay Protocol Configuration parameters.
FrDemCtrlTestResultPtr	Reference to the DEM configuration structure which contains ID and state (disabled/enabled) of reported diagnostic message.

3.6.2.1.1 Example for Fr_ConfigurationType

Example configuration consists of two configurations for two variant configuration VS0 and VS1 for one CC and two CCs (if available on the hardware) controlled by one FlexRay driver, therefore two instances of configuration arrays exist as shown below.

```
/* This structure puts together all information about controllers for Fr_Config_0_VS0
configuration */
CONST(Fr_ConfigurationType, FR_APPL_CONST) Fr_Config_0_VS0[] =
{
    {&FrMC0_Ctrl0_CCHardwareCfgSet_PB, &FrIfMC0_Clst0_Ctrl0_BufferCfgSet_PB,
&FrMC0_Ctrl0_CCLowLevelCfgSet_PB, &FrMC0_Ctrl0_CCReadBackParamSet_PB[0],
&FrMC0_Ctrl0_DemEventParameter_PB}, /* Fr_Controller 0 */
    {NULL_PTR, NULL_PTR, NULL_PTR, NULL_PTR, NULL_PTR}
};
```

```
/* This structure puts together all information about controllers for Fr_Config_1_VS1
configuration */
CONST(Fr_ConfigurationType, FR_APPL_CONST) Fr_Config_1_VS1[] =
{
    {&FrMC0_Ctrl0_CCHardwareCfgSet_PB, &FrIfMC1_Clst0_Ctrl0_BufferCfgSet_PB,
&FrMC1_Ctrl0_CCLowLevelCfgSet_PB, &FrMC1_Ctrl0_CCReadBackParamSet_PB[0],
&FrMC1_Ctrl0_DemEventParameter_PB}, /* Fr_Controller 0 */
    {&FrMC0_Ctrl1_CCHardwareCfgSet_PB, &FrIfMC1_Clst0_Ctrl1_BufferCfgSet_PB,
&FrMC1_Ctrl1_CCLowLevelCfgSet_PB, &FrMC1_Ctrl1_CCReadBackParamSet_PB[0],
&FrMC1_Ctrl1_DemEventParameter_PB}, /* Fr_Controller 1 */
    {NULL_PTR, NULL_PTR, NULL_PTR, NULL_PTR, NULL_PTR}
};
```

Example of usage:

```
/* Declare the Configuration Structure - from Fr_PBcfg_VS0.h and Fr_PBcfg_VS1.h file */
extern CONST(Fr_ConfigurationType, FR_CONST) (Fr_Config_0_VS0[]);
extern CONST(Fr_ConfigurationType, FR_CONST) (Fr_Config_1_VS1[]);

/* CONFIGURATION VS0 */
/* Initialize the FlexRay CC */
  Fr_Init((void*)(&Fr_Config_0_VS0));
/* Configure low level, hardware, message buffer and timer parameters */
  Fr_ControllerInit(0);
  Fr_ControllerInit(1);

/* CONFIGURATION VS1 */
/* Initialize the FlexRay CC */
  Fr_Init((void*)(&Fr_Config_1_VS1));
/* Configure low level, hardware, message buffer and timer parameters */
```

```
  Fr_ControllerInit(0);
  Fr_ControllerInit(1);
```

3.6.2.2 Structure Fr_CCHardwareConfigType

The structure of the type `Fr_CCHardwareConfigType` contains the FlexRay module specific parameters. The instance of the `Fr_CCHardwareConfigType` structure is used by the `Fr_ControllerInit()` function.

Declaration

```
typedef struct
{
  CONST(uint32, FR_CONST) CCBaseAddress;
  CONST(uint32, FR_CONST) CCFlexRayMemoryBaseAddress;
  CONST(Fr_ChannelType, FR_CONST) Channels;
  CONST(boolean, FR_CONST) SingleChannelModeEnabled;
  CONST(Fr_CCClockSourceType, FR_CONST) ClockSource;
  CONST(uint8, FR_CONST) Bitrate;
  CONST(uint8, FR_CONST) Timeout;
  CONST(uint16, FR_CONST) SyncFrameTableOffset;
  CONST(boolean, FR_CONST) EnableSecondaryAbsTimer;
} Fr_CCHardwareConfigType;
```

Table 3-50. Structure Fr_CCHardwareConfigType member description

Member	Description
CCBaseAddress	The base address of the FlexRay module. The address depends on selected MCU.
CCFlexRayMemoryBaseAddress	The base address of the FlexRay memory within the system memory. Value is assigned via compiler/linker command file. **CAUTION:** The FlexRay memory base address must be aligned to the 16-byte boundary.
Channels	The value is equivalent to pChannels parameter in the instance of the type `Fr_CCLowLevelConfigSetType`.
SingleChannelModeEnabled	Enabling of single channel mode. The value FALSE represents the dual channel mode. Single channel mode supports devices that have only one FlexRay port available. This port can be connected to either the physical bus channel A or the physical bus channel B.
ClockSource	Protocol engine clock source select. The following options are available: FR_CLK_SRC_PLL for the internal PLL or FR_CLK_SRC_CRYSTAL_OSCILLATOR for the external oscillator.
Bitrate	Required FlexRay communication bit rate. This value will be loaded into the MCR register during `Fr_Init()` function call.
Timeout	Timeout value in the SYMATOR register. The timeout value corresponds directly to acceptable number of wait states on the system bus. It has to be lest than or equal (0.45*f_chi)-8, where f_chi is the frequency of the CHI clock.
SyncFrameTableOffset	This field configures the offset of the Sync Frame Tables in the FlexRay memory area.
EnableSecondaryAbsTimer	Configure timer T2 to be an absolute timer.

3.6.2.2.1 Example for Fr_CCHardwareConfigType

The base address for the MPC574XG controller is 0xffe5000U as shows the example.

```
CONST(Fr_CCHardwareConfigType, FR_APPL_CONST) FrMC0_CCHardwareCfgSet_PB =
{
  0xffe5000U, /* Controller base address */
  FR_CC_FLEXRAY_MEMORY_BASE_ADDR, /* FlexRay memory base address */
  FR_CHANNEL_AB, /* FlexRay channels */
  FALSE,  /* FALSE = Dual channel mode selected */
  FR_CLK_SRC_PLL, /* FlexRay protocol engine clock source */
  0U,  /* Bus speed: 10 Mb/s */
  10U, /* System memory access timeout */
  420U, /* SFTOR register value */
  FALSE /* The second absolute timer is disabled */
};
```

Example of usage:

```
/* The CC is configured - message buffers, low level parameters and hardware
dependent parameters */
  Fr_ControllerInit(0);
```

Please refer to the MPC5748G Reference Manual [Table 2-3] for more details.

3.6.2.3 Structure Fr_CCLowLevelConfigSetType

The structure of the type `Fr_CCLowLevelConfigSetType` contains the FlexRay protocol specific parameters. The configuration information corresponds with the contents of the FlexRay module Protocol Configuration Registers (registers PCR0 to PCR30), the FlexRay Protocol Specification, the AUTOSAR FlexRay Driver and the FlexRay Interface specifications. The FlexRay Driver uses these values for Protocol Configuration Registers initialization and for internal operations.

Declaration

```
typedef struct
{
  CONST(uint32, FR_CONST)  RegPCR0_1;
  CONST(uint32, FR_CONST)  RegPCR2_3;
  CONST(uint32, FR_CONST)  RegPCR4_5;
  CONST(uint32, FR_CONST)  RegPCR6_7;
  CONST(uint32, FR_CONST)  RegPCR8_9;
  CONST(uint32, FR_CONST)  RegPCR10_11
  CONST(uint32, FR_CONST)  RegPCR12_13
  CONST(uint32, FR_CONST)  RegPCR14_15
  CONST(uint32, FR_CONST)  RegPCR16_17
  CONST(uint32, FR_CONST)  RegPCR18_19
  CONST(uint32, FR_CONST)  RegPCR20_21
  CONST(uint32, FR_CONST)  RegPCR22_23
  CONST(uint32, FR_CONST)  RegPCR24_25
  CONST(uint32, FR_CONST)  RegPCR26_27
  CONST(uint32, FR_CONST)  RegPCR28_29
  CONST(uint16, FR_CONST)  RegPCR30
  CONST(uint16, FR_CONST)  gNumberOfStaticSlots;
  CONST(uint8, FR_CONST)   gNetworkManagementVectorLength;
  CONST(uint8, FR_CONST)   pPayloadLengthDynMax;
  CONST(uint8, FR_CONST)   gPayloadLengthStatic;
} Fr_CCLowLevelConfigSetType;
```

Table 3-51. Structure Fr_CCLowLevelConfigSetType member description

Member	Description
RegPCR0_1	Configures Protocol Configuration Register 0_1 value.
RegPCR2_3	Configures Protocol Configuration Register 2_3 value.
RegPCR4_5	Configures Protocol Configuration Register 4_5 value.
RegPCR6_7	Configures Protocol Configuration Register 6_7 value.
RegPCR8_9	Configures Protocol Configuration Register 8_9 value.
RegPCR10_11	Configures Protocol Configuration Register 10_11 value.
RegPCR12_13	Configures Protocol Configuration Register 12_13 value.
RegPCR14_15	Configures Protocol Configuration Register 14_15 value.
RegPCR16_17	Configures Protocol Configuration Register 16_17 value.
RegPCR18_19	Configures Protocol Configuration Register 18_19 value.
RegPCR20_21	Configures Protocol Configuration Register 20_21 value.
RegPCR22_23	Configures Protocol Configuration Register 22_23 value.
RegPCR24_25	Configures Protocol Configuration Register 24_25 value.
RegPCR26_27	Configures Protocol Configuration Register 26_27 value.
RegPCR28_29	Configures Protocol Configuration Register 28_29 value.
RegPCR30	Configures Protocol Configuration Register 30 value.
gNumberOfStaticSlots	Configures gNumberoftaticslots parameter.
gNetworkManagementVectorLength	Configures gNetworkManagementVectorLength parameter.
pPayloadLengthDynMax	Configures pPayloadLengthDynMax parameter.
gPayloadLengthStatic	Configures gPayloadLengthStatic parameter.

3.6.2.3.1 Example for Fr_CCLowLevelConfigSetType

Example below shows possible values for structure of Fr_CCLowLevelConfigSetType.

```
CONST(Fr_CCLowLevelConfigSetType, FR_APPL_CONST) FrMC0_CCLowLevelCfgSet_PB =
{
    0x14411347U,            /*  PCR0_1 */
    0x083cdc4aU,            /*  PCR2_3 */
    0xb52dbf3bU,            /*  PCR4_5 */
    0x06081314U,            /*  PCR6_7 */
    0xeab4c078U,            /*  PCR8_9 */
    0x1388d338U,            /*  PCR10_11  PCR11 - pOffsetCorrectionStart from spec. 3.0 used
instead of gOffsetCorrectionStart from spec. 2.1*/
    0x06c0143aU,            /*  PCR12_13  PCR12 - Key slot crc is 1728 = 0x06c0 */
    0x78461f31U,            /*  PCR14_15 */
    0x100c3e63U,            /*  PCR16_17 */
    0xfc0a1310U,            /*  PCR18_19  PCR18 - Key Slot ID is 10 */
    0x18180006U,            /*  PCR20_21 */
    0x06e30d40U,            /*  PCR22_23 */
    0x08830ae7U,            /*  PCR24_25 */
    0x86e30f99U,            /*  PCR26_27 */
    0x405000a2U,            /*  PCR28_29 */
    0x0005U,            /*  PCR30 */
    60U,                /*  gNumberOfStaticSlots */
    2U,                 /*  gNetworkManagementVectorLength */
    8U,                 /*  pPayloadLengthDynMax */
    16U                 /*  gPayloadLengthStatic */
};
```

Details about FlexRay related configuration parameters and the allowed parameter ranges can be found in FlexRay module documentation in MPC5748G Reference Manual [Table 2-3], FlexRay Protocol Specification and AUTOSAR FlexRay Driver software specification document [Table 2-3].

Example of usage:

```
/* The FlexRay CC is configured - message buffers, low level parameters and
hardware dependent parameters */
  Fr_ControllerInit(0);
```

3.6.2.4 FlexRay Timers

The FlexRay module provides two timers, T1 and T2, which run on the FlexRay time base. Each of these timers generates a maskable interrupt when it reaches a configured point in time. The timer T1 is an absolute timer. The timer T2 can be configured as an absolute or a relative timer.

Note

FlexRay ASR4.2 Specification does not provide an API to support a relative timer. The timer T2 can be configured as an absolute timer only.

3.6.2.5 Error Codes

Table 3-52. Error code 001 description

Error Code	001
Description	Referenced Fr CC ("FrIfFrCtrlRef" parameter) in the FrIf Controller configuration "[FrIfCC_configuration_path]" is not valid.
Solution	Correct value of given node to contain a valid reference to the Fr CC node.

Table 3-53. Error code 002 description

Error Code	002
Description	CC [x] Configuration "name" - Fr Controller is assigned to more than one FrIf cluster in one FrIf (multiple) configuration.

Table continues on the next page...

Table 3-53. Error code 002 description (continued)

Solution	Assign the controller identified by number x to only one cluster.

Table 3-54. Error code 003 description

Error Code	003
Description	The FrIfController is not used in any FrIfCluster.
Solution	Check that the FrIfCluster container contains at least one FrIfController subcontainer.

Table 3-55. Error code 004 description

Error Code	004
Description	FrKeySlotId value is greater than the FrIfGNumberOfStaticSlots value.
Solution	Check whether the FrKeySlotId parameter is correctly configured to a slot in the static segment.

Table 3-56. Error code 005 description

Error Code	005
Description	Setting the Startup bit and not setting the Sync bit leads to an invalid configuration!
Solution	Check that FrPKeySlotUsedForSync parameter is not set to false in case of FrPKeySlotUsedForStartup is set to true.

Table 3-57. Error code 006 description

Error Code	006
Description	FrIfConfig/FrIfCluster/FrIfController/FrIfFrameTriggering/FrIfFrameStructureRef contains an invalid reference or refence is missing.
Solution	Check whether reference to the FrIfFrameStructure is valid.

Table 3-58. Error code 007 description

Error Code	007
Description	Selected Bit Rate value is not correctly configured.
Solution	Check and correct ChannelBitrate parameter in VendorSpecific containers according the Microcontroller Reference Manual (see MPC5748G Reference Manual [Table 2-3]).

Table 3-59. Error code 008 description

Error Code	008
Description	The list of FrAbsoluteTimer does not contain any Absolute timer, at least one is required.
Solution	Check number of configured absolute counters and their indices validity

Table 3-60. Error code 009 description

Error Code	009
Description	Too many configured absolute timers. The maximum number of absolute timers is 2.
Solution	Configure at most two timers.

Table 3-61. Error code 010 description

Error Code	010
Description	Configured absolute timer has an invalid FrAbsTimerIdx.
Solution	Configure FrAbsTimerIdx index of each relative timer to value 0 or 1.

Table 3-62. Error code 011 description

Error Code	011
Description	The index of each absolute timer should be lower than number of absolute timers in list of absolut timers (FrAbsoluteTimer).
Solution	Check whether the index is within range 0 to number of absolut timers.

Table 3-63. Error code 012 description

Error Code	012
Description	FrIfFrameTriggering reference in the “[FrIfLpdu_path]” node is not valid.
Solution	Correct value of given node to contain a valid reference in the “[FrIfLpdu_path]”.

Table 3-64. Error code 013 description

Error Code	013
Description	List of FIFOs (FrFrFifo in the FrController container) contains more than 2 FIFOs, only 2 of them are supported by the hardware.
Solution	Check that count of FIFOs is less or equal to 2.

Table 3-65. Error code 014 description

Error Code	014
Description	Both FIFOs are configured for the same channel
Solution	Check the configuration, only one FIFO is available for one channel.

Table 3-66. Error code 015 description

Error Code	015
Description	List of range (FrRange in the FrFifo container) does not contain any range filter.
Solution	Check that at least one range is defined.

Table 3-67. Error code 016 description

Error Code	016
Description	List of range (FrRange in the FrFifo container) contains more than 4 range filters.
Solution	Check that at most 4 ranges are defined.

Table 3-68. Error code 017 description

Error Code	017
Description	Parameter FrIfLSduLength is not equal to 2*FrIfGPayloadLengthStatic although the slot is configured in static segment.
Solution	Check the lengths of parameters FrIfLSduLength(in the FrIfFrameTriggering container) and FrIfGPayloadLengthStatic(in the FrIfCluster container).

Table 3-69. Error code 018 description

Error Code	018
Description	The FrIfAllowDynamicLSduLength parameter should not be configured for a frame in static segment of communication cycle.
Solution	Disable FrIfAllowDynamicLSduLength parameter for a frame configured for static segment of communication cycle.

Table 3-70. Error code 019 description

Error Code	019
Description	Parameter FrIfLSduLength is greater than 2*FrPPayloadLengthDynMax.
Solution	Check the lengths of parameters FrIfLSduLength(in the FrIfFrameTriggering container) and FrPPayloadLengthDynMax(in the FrIfCluster container).

Table 3-71. Error code 020 description

Error Code	020
Description	Some index of LPdus (FrIfLPduIdx in FrIfLPdu container) is not unique within all LPdus.
Solution	Check that all LPdu indices are unique.

Table 3-72. Error code 021 description

Error Code	021
Description	LPdu indices do not create continuous row beginning with 0.
Solution	Check that all LPdu indices constitute continuous row beginning with 0

Table 3-73. Error code 022 description

Error Code	022

Table continues on the next page...

Table 3-73. Error code 022 description (continued)

Description	FrIfChannel is set to FRIF_CHANNEL_B(or FR_IF_CHANNEL_AB) but the controller is configured as single channel device. For a single channel device, the application can access and configure only the registers related to internal channel A.
Solution	If single channel mode is selected, configure FRIF_CHANNEL_A.

Table 3-74. Error code 023 description

Error Code	023
Description	The SingleChannelModeEnabled is enabled but FrPWakeupChannel is set to FR_CHANNEL_B. For a single channel device, the application can access and configure only the registers related to internal channel A.
Solution	If SingleChannelModeEnabled parameter is enabled, configure FrPWakeupChannel to FR_CHANNEL_A.

Table 3-75. Error code 024 description

Error Code	024
Description	The FrPChannels is set to FR_CHANNEL_A(FR_CHANNEL_B) only but FrPWakeupChannel is set to FR_CHANNEL_B(FR_CHANNEL_A).
Solution	Modify the FrPWakeupChannel parameter to have the same value as the FrPChannels parameter.

Table 3-76. Error code 025 description

Error Code	025
Description	FrIfChannel is set to FRIF_CHANNEL_B but the controller is configured as connected to FR_CHANNEL_A only.
Solution	Configure the FrIfChannel parameter to FRIF_CHANNEL_A or check the FrPChannels parameter.

Table 3-77. Error code 026 description

Error Code	026
Description	FrIfChannel is set to FRIF_CHANNEL_A but the controller is configured as connected to FR_CHANNEL_B only.
Solution	Configure the FrIfChannel parameter to FRIF_CHANNEL_B or check the FrPChannels parameter.

Table 3-78. Error code 027 description

Error Code	027
Description	The SingleChannelModeEnabled parameter is enabled but FrPChannels is set to FR_CHANNEL_AB.
Solution	Check whether the controller channels are correctly configured (parameters SingleChannelModeEnabled and FrPChannels).

Table 3-79. Error code 028 description

Error Code	028
Description	FrRangeMax is lower than FrRangeMin.
Solution	Verify range configuration.

Table 3-80. Error code 029 description

Error Code	029
Description	LPdu is configured for dynamic segment but FrIfReconfigurable is set.
Solution	Check that reconfiguration is disabled for dynamic segment.

Table 3-81. Error code 030 description

Error Code	030
Description	FrRange does not contain any acceptance range.
Solution	Check that at least one acceptance range is defined.

Table 3-82. Error code 031 description

Error Code	031
Description	LPdu will be received into the FIFO A but the FrIfReconfigurable is set.
Solution	Check which LPdus are received into the FIFO A and check that they are not reconfigurable.

Table 3-83. Error code 032 description

Error Code	032
Description	LPdu will be received into the FIFO B but the FrIfReconfigurable is set.
Solution	Check which LPdus are received into the FIFO B and check that they are not reconfigurable.

Table 3-84. Error code 033 description

Error Code	033
Description	FrIfCommunicationOperationIdx is not unique.
Solution	Check that all Communication Operation indices (FrIfCommunicationOperationIdx) are unique, separately for each Job.

Table 3-85. Error code 034 description

Error Code	034
Description	Not enough resources for all LPdus even after reconfiguration.
Solution	Consider the possibility of reconfiguration for another LPdus or reduce number of LPdus.

Table 3-86. Error code 035 description

Error Code	035
Description	Not enough resources for all LPdus.
Solution	Try enable the reconfiguration (ForceReconfiguration for global reconfiguration enable and FrIfReconfigurable for each different LPdu)

Table 3-87. Error code 036 description

Error Code	036
Description	Parameter FrIfCycleRepetition contains unsupported value.
Solution	Verify the FrIfCycleRepetition parameter (possible values: 1,2,4,8,16,32,64).

Table 3-88. Error code 037 description

Error Code	037
Description	FrControllerDemEventParameterRefs/FrDemCtrlTestResultRef has no reference to DemEventID
Solution	Select reference to DEM plugin for FrControllerDemEventParameterRefs/FrDemCtrlTestResultRef parameter.

Table 3-89. Error code 038 description

Error Code	038
Description	FrIfChannel is set to FRIF_CHANNEL_AB but the controller is configured as connected to FR_CHANNEL_B only. Transmission or reception in the dynamic segment will not work.
Solution	In the dynamic segment CC use only channel A in case of both channels are configured. Configure FrIfChannels to FRIF_CHANNEL_B only.

Table 3-90. Error code 039 description

Error Code	039
Description	SingleChannelModeEnabled is enabled but the FrFifo container contains more than 1 FIFO, only 1 FIFO is supported when the controller is configured as single channel device.
Solution	Reduce number of FIFOs in the FrFifo container to one FIFO only.

Table 3-91. Error code 040 description

Error Code	040
Description	SingleChannelModeEnabled is enabled but FrChannels is set to FR_CHANNEL_B. For a single channel device, the application can access and configure only the registers related to internal channel A.
Solution	Configure the FrChannels parameter in the FrFifo container to FR_CHANNEL_A.

Table 3-92. Error code 041 description

Error Code	041
Description	The FrFifo container contains more than 1 FIFO but FrPChannels is set to FR_CHANNEL_A(FR_CHANNEL_B) only. Only one FIFO per channel is supported by the hardware.
Solution	Reduce number of FIFOs in the FrFifo container to one FIFO only.

Table 3-93. Error code 042 description

Error Code	042
Description	Wrong channel is assigned to FIFO. FrChannels is configured to FR_CHANNEL_A(FR_CHANNEL_B) but FrPChannels is configured to FR_CHANNEL_B(FR_CHANNEL_A).
Solution	Configure the FrChannels parameter in the FrFifo container to the same value as is configured in FrPChannels.

Table 3-94. Error code 043 description

Error Code	043
Description	Maximum FIFO depth 255 entries was exceeded!.
Solution	Reduce FrFifoDepth either for FIFO A or FIFO B. **Note:** FIFO depth can be configured up to 255 entries for each FIFO. In case of both FIFO A and FIFO B are configured total FIFO depth can not exceed 255 entries.

Table 3-95. Error code 044 description

Error Code	044
Description	Some index of CC (FrCtrlIdx in the FrController container) is not unique.
Solution	Check that all CC indices are unique.

Table 3-96. Error code 045 description

Error Code	045
Description	CC indices (FrCtrlIdx in the FrController container) do not create continuous row beginning with 0.
Solution	Check that all CC indices constitute continuous row beginning with 0.

Table 3-97. Error code 046 description

Error Code	046
Description	FrIfClstIdx indices do not create zero-based consecutive sequence within the listed container.
Solution	Check that all FrIfClstIdx indices constitute continuous sequence beginning with 0.

Table 3-98. Error code 047 description

Error Code	047
Description	FrIfCtrlIdx indices do not create zero-based consecutive sequence within the listed container.
Solution	Check that all FrIfCtrlIdx indices constitute continuous sequence beginning with 0.

Table 3-99. Error code 5xx description

Error Code	5xx
Description	Internal error
Solution	If this error is shown, some unexpected situation occurred. Please contact support with following information: • Tresos Studio version • MCAL release version (best to send concrete Fr and FrIf plugins) • Error number and description • FrIf.xdm and Fr.xdm (located in the project\config folder)

Table 3-100. Error code 800 description

Error Code	800
Description	The length of gdNit must be equal to [gMacroPerCycle - gdSymbolWindow - (gNumberOfStaticSlots * gdStaticSlot) - (gNumberOfMinislots * gdMinislot)].
Solution	Verify configuration prameters: gdNit, gMacroPerCycle, gdSymbolWindow, gNumberOfStaticSlots, gdStaticSlot, gNumberOfMinislots, gdMinislot.

3.6.2.5.1 Structure Fr_CCBufferConfigSetType

The instance of the `Fr_CCBufferConfigSetType` structure is used by the `Fr_ControllerInit()`, `Fr_TransmitTxLPdu()`, `Fr_PrepareLPdu()`, `Fr_ReceiveRxLPdu()` and `Fr_CheckTxLPduStatus()` functions to access the message buffers configuration.

Declaration

```
typedef struct
{
  CONSTP2CONST(Fr_CCLPduInfoType, AUTOMATIC, FR_APPL_CONST) LPduInfoPtr;
  CONSTP2CONST(Fr_CCBufferAddress32Type, AUTOMATIC, FR_APPL_CONST) BufferAddressTable;
  CONSTP2CONST(Fr_CCBufferOffset16Type, AUTOMATIC, FR_APPL_CONST) BufferOffsetTable;
  CONST(uint16, FR_CONST) BuffersConfiguredCount;
  CONST(uint8, FR_CONST) MessageBufferSegment1DataSize;
  CONST(uint8, FR_CONST) MessageBufferSegment2DataSize;
  CONST(uint8, FR_CONST) LastMBSEG1;
  CONST(uint8, FR_CONST) LastMBUTIL;
  CONST(uint8, FR_CONST) RSBIR_A1BufferIndexInit;
  CONST(uint8 FR_CONST) RSBIR_B1BufferIndexInit;
  CONST(uint8, FR_CONST) RSBIR_A2BufferIndexInit;
  CONST(uint8, FR_CONST) RSBIR_B2BufferIndexInit;
} Fr_CCBufferConfigSetType;
```

Table 3-101. Structure Fr_CCBufferConfigSetType member description

Member	Description
LPduInfoPtr	Reference to configuration information of one message buffer configuration set.
BufferAddressTable	Reference to MB address table. This table contains contains complete 32 bit message buffer data area addresses (message buffer data offset added to memory base address).
BufferOffsetTable	Reference to MB offset table. This table contains contains only 16 bit message buffer data area offsets (without memory base address).
BuffersConfiguredCount	The number of items in the configuration structure of the type `Fr_CCLPduInfoType`. It is the number of LPdu's processed by the corresponding CC. For instance the value 1 represents one configured LPdu.
MessageBufferSegment1DataSize	Message buffer segment 1 data size [Words].
MessageBufferSegment2DataSize	Message buffer segment 2 data size [Words].
LastMBSEG1	The message buffer number of the last individual message buffer that is assigned to the first message buffer segment. (The number of MB's in the first segment - 1).
LastMBUTIL	The message buffer number of the last utilized individual message buffer. (The number of MB's in the first segment + the number of MB's in the second segment - 1).
RSBIR_A1BufferIndexInit	Defines initial message buffer header index of receive shadow buffer for channel A and segment 1.
RSBIR_A2BufferIndexInit	Defines initial message buffer header index of receive shadow buffer for channel A and segment 2.
RSBIR_B1BufferIndexInit	Defines initial message buffer header index of receive shadow buffer for channel B and segment 1.
RSBIR_B2BufferIndexInit	Defines initial message buffer header index of receive shadow buffer for channel B and segment 2.

3.6.2.5.1.1 Example for Fr_CCBufferConfigSetType

In this example, three LPdu's are configured. All message buffers configured for transmission or reception in the static part of the communication cycle belong to the first segment (segment 1 and both channels), one message buffer configured for reception in the dynamic part of the communication cycle belongs to the second segment (segment 2 and channel A).

The layout of the segments and message buffer is as follows:

- Segment 1 is located in the message buffer number range of 0 to 1 (the number of message buffers in segment 1 is 2)
- Segment 2 from 2 to 2 (the number of message buffers in segment 2 is 1).

The initial message buffer header index allocation is as follows:

- The first shadow buffer for segment 1 and channel A - initial header index is 3.
- The second shadow buffer for segment 1 and channel B - initial header index is 4.

- The third shadow buffer for segment 2 and channel A - initial header index is 5.
- The fourth shadow buffer for segment 2 and channel B - is not configured, the initial header index value can be arbitrary.

```
CONST(Fr_CCBufferConfigSetType, FR_APPL_CONST) FrIfMC0_Clst0_Ctrl0_BufferCfgSet_PB =
{
  &FrIfMC0_Clst0_Ctrl0_LPduInfoCfgSet_PB[0],  /* LPdu configuration set */
  &FrIfMC0_Clst0_Ctrl0_BAddressTable_PB[0], /* Buffer Addresses Table */
  &FrIfMC0_Clst0_Ctrl0_BOffset16Table_PB[0],  /* Buffer Offsets Table */
  3U, /* Number of items in FrIfMC0_Clst0_Ctrl0_LPduInfoCfgSet_PB */
  16U, /* Data size in buffers segment 1 - gPayloadLengthStatic */
  8U, /* Data size in buffers segment 2 - pPayloadLengthDynMax */
  1U, /* Last MB in segment 1 (Number of MB in Segment1 - 1) */
  2U, /* Last individual MB; (Number of MB in Segment1 + Number of MB in Segment2 - 1) */
  /* Receive shadow buffers configuration */
  3U, /* Ch A, seg 1 - the initial index of the MB header field */
  4U, /* Ch B, seg 1 - the initial index of the MB header field */
  5U, /* Ch A, seg 2 - the initial index of the MB header field */
  6U /* Ch B, seg 2 - unused */
};
```

Example of usage:

```
  Fr_ControllerInit(0);
```

3.6.2.5.2 Structure Fr_CCTxBufferConfigType

The configuration information for each individual transmit message buffer is stored in a separate instance of the type `Fr_CCTxBufferConfigType`. The address of each separate structure of the type `Fr_CCTxBufferConfigType` should be passed into the relevant BufferConfigPtr parameter of the configuration structure of the type `Fr_CCLPduInfoType`.

The instance of the `Fr_CCTxBufferConfigType` structure is used by the `Fr_ControllerInit()`, `Fr_TransmitTxLPdu()`, `Fr_PrepareLPdu()`, and `Fr_CheckTxLPduStatus()`.

Declaration

```
typedef struct
{
  CONST(uint16, FR_CONST) TxFrameID;
  CONST(uint16, FR_CONST) HeaderCRC;
  CONST(uint8, FR_CONST) TxPayloadLength;
  CONST(boolean, FR_CONST) TxChannelAEnable;
  CONST(boolean, FR_CONST) TxChannelBEnable;
  CONST(boolean, FR_CONST) PayloadPreamble;
  CONST(boolean, FR_CONST) TxCycleCounterFilterEnable;
  CONST(uint8, FR_CONST) TxCycleCounterFilterValue;
  CONST(uint8, FR_CONST) TxCycleCounterFilterMask;
  CONST(boolean, FR_CONST) AllowDynamicLength;
} Fr_CCTxBufferConfigType;
```

Table 3-102. Structure Fr_CCTxBufferConfigType member description

Member	Description
TxFrameID	The slot in which the frame will be transmitted.
HeaderCRC	Header CRC of this frame.
TxPayloadLength	Defines the Payload length of the transmitted frame [Words].
TxChannelAEnable	Defines whether the channel A is used for the transmission.
TxChannelBEnable	Defines whether the channel B is used for the transmission.
PayloadPreamble	The payload preamble indicator • Disabled: No network management vector or message ID in the frame payload data. • Enabled: • in the Static Segment: Frame payload data contains network management vector • in the Dynamic Segment: Frame payload data contains message ID.
TxCycleCounterFilterEnable	Defines whether or not the cycle counter filtering is enabled. If the Transmit cycle filter is enabled, the frame is transmitted only if cycle AND TxCycleCounterFilterMask is equal to TxCycleCounterFilterValue AND TxCycleCounterFilterMask **Note:** It is possible to configure more transmit buffers with different filters to one slot, however all of them must have same channel assignment. Even more, it is possible to multiplex transmit and receive buffers for one slot in the dynamic segment (means only in the dynamic segment).
TxCycleCounterFilterValue	Defines the filter value for the cycle counter filtering.
TxCycleCounterFilterMask	Defines the filter mask for cycle counter filtering.
AllowDynamicLength	The dynamic payload length indicator • Disabled: Dynamic payload length is not supported for transmission resource. • Enabled: Dynamic payload length is supported for transmission resource and payload length can be reconfigured. This possibility is available only for transmission resource configured for dynamic segment.

3.6.2.5.2.1 Example for Fr_CCTxBufferConfigType

The transmit message buffer is configured to transmit frames on both channels in the 2nd slot. Cycle counter filter is disabled. The payload length is 16 words. No network management vector is enabled. Dynamic payload length is not enabled (message buffer is configured for static segment).

```
CONST(Fr_CCTxBufferConfigType, FR_APPL_CONST) FrIfMC0_Clst0_Ctrl0_LPdu0_Cfg_PB =
{
  2U, /* Transmit Frame ID, it is the Key Slot */
  0x02ed, /* Frame Header CRC */
  16U, /* Data Length in Words */
  TRUE,   /* Reception on channel A enabled */
  TRUE,   /* Reception on channel B enabled */
  FALSE,  /* Payload preamble disabled */
  FALSE,  /* Cycle counter filtering disabled */
  0U, /* Cycle counter filter match value */
  0x0000U, /* Cycle counter filter mask (repetition each 1 cycles) */
  FALSE,  /* Dynamic payload length disabled */
};
```

```
CONST(Fr_CCLPduInfoType, FR_APPL_CONST) FrIfMC0_Clst0_Ctrl0_LPduInfoCfgSet_PB[] =
{
  {FR_TRANSMIT_BUFFER, & FrIfMC0_Clst0_Ctrl0_LPdu0_Cfg_PB, 2U, TRUE, FALSE},
  {FR_TRANSMIT_BUFFER, & FrIfMC0_Clst0_Ctrl0_LPdu1_Cfg_PB, 3U, FALSE, FALSE},
};
```

Example of usage: Only the message buffer 2 is configured.

```
uint8 Data[32] = {0};
/* The LPdu with the index 0 is configured */
  Fr_ControllerInit(0);
  Fr_TransmitTxLPdu(0, 0, &Data[0], 32U);
```

3.6.2.5.3 Structure Fr_CCRxBufferConfigType

The configuration information for each individual receive message buffer is stored in a separate instance of the type `Fr_CCRxBufferConfigType`.

The address of each separate structure of the type `Fr_CCRxBufferConfigType` should be passed into the relevant BufferConfigPtr parameter of the configuration structure of the type `Fr_CCLPduInfoType`.

The instance of the `Fr_CCRxBufferConfigType` structure is used by the `Fr_ControllerInit()`, `Fr_ReceiveRxLPdu()` and `Fr_PrepareLPdu()` functions.

Note

The receive shadow buffers have to be configured together with the receive message buffers for correct FlexRay module operation in an instance of the `Fr_CCBufferConfigSetType` type.

Declaration

```
typedef struct
{
  CONST(uint16, FR_CONST) RxFrameID;
  CONST(uint8, FR_CONST) RxPayloadLength;
  CONST(boolean, FR_CONST) RxChannelAEnable;
  CONST(boolean, FR_CONST) RxChannelBEnable;
  CONST(boolean, FR_CONST) RxCycleCounterFilterEnable;
  CONST(uint8, FR_CONST) RxCycleCounterFilterValue;
  CONST(uint8, FR_CONST) RxCycleCounterFilterMask;
} Fr_CCRxBufferConfigType;
```

Table 3-103. Structure Fr_CCRxBufferConfigType member description

Member	Description
RxFrameID	A filter value to determine whether or not the message buffer is used for reception of a message received in a slot whose slot ID equals the Frame ID (FID).
RxPayloadLength	Maximum configured payload length in [Words].
RxChannelAEnable	Defines whether the channel A is used for the reception.
RxChannelBEnable	Defines whether the channel B is used for the reception.
RxCycleCounterFilterEnable	CDefines whether or not the cycle counter filtering is enabled. If the receive cycle filter is enabled, the frame is received to this message buffer only if cycle AND RxCycleCounterFilterMask is equal to RxCycleCounterFilterValue AND RxCycleCounterFilterMask.
RxCycleCounterFilterValue	Defines the filter value for the cycle counter filtering.
RxCycleCounterFilterMask	Defines the filter mask for cycle counter filtering.

3.6.2.5.3.1 Example for Fr_CCRxBufferConfigType

The receive message buffer is configured to receive frames on channel A in the 3rd slot. Cycle counter filter is disabled. The maximal payload length is 16 words.

```
CONST(Fr_CCRxBufferConfigType, FR_APPL_CONST) FrIfMC0_Clst0_Ctrl0_LPdu1_Cfg_PB =
{
  3U, /* Receive Frame ID */
  16U, /* Data Length in Words */
  TRUE,   /* Reception on channel A enabled */
  TRUE,   /* Reception on channel B enabled */
  FALSE,  /* Cycle counter filtering disabled */
  0U, /* Cycle counter filter match value */
  0x0000U /* Cycle counter filter mask (repetition each 1 cycles) */
};
```

```
CONST(Fr_CCLPduInfoType, FR_APPL_CONST) FrIfMC0_Clst0_Ctrl0_LPduInfoCfgSet_PB[] =
{
  {FR_TRANSMIT_BUFFER, & FrIfMC0_Clst0_Ctrl0_LPdu0_Cfg_PB, 0U, TRUE, FALSE},
  {FR_RECEIVE_BUFFER, & FrIfMC0_Clst0_Ctrl0_LPdu1_Cfg_PB, 1U, TRUE, FALSE},
  {FR_TRANSMIT_BUFFER, & FrIfMC0_Clst0_Ctrl0_LPdu2_Cfg_PB, 1U, FALSE, FALSE},
};
```

Example of usage: The message buffers 0 and 1 are configured.

```
uint8 RxData[32] = {0};
uint8 RxLength = 0;
Fr_RxLPduStatusType RxStatus;
/* The LPdu's with the indexes 0 and 1 are configured */
  Fr_ControllerInit(0);
  Fr_ReceiveRxLPdu(0U, 2U, &RxData[0], &RxStatus, &RxLength);
```

3.6.2.5.4 Structure Fr_CCFIFOConfigType

The configuration information for each FIFO storage is stored in a separate instance of the type `Fr_CCFIFOConfigType`. The address of each separate structure of the type `Fr_CCFIFOConfigType` should be passed into the relevant BufferConfigPtr parameter of the configuration structure of the type `Fr_CCLPduInfoType`.

Declaration

```
typedef struct
{
    CONST(Fr_CCFIFOChannelType, FR_CONST) FIFOChannel;
    CONST(uint16, FR_CONST) FIFOStartIndex;
    CONST(uint8, FR_CONST) FIFODepth;
    CONST(uint8, FR_CONST) FIFOEntrySize;
    CONST(uint8, FR_CONST) MessageIDFilterMask;
    CONST(uint8, FR_CONST) MessageIDFilterMatch;
    CONST(Fr_CCFIFORangeFiltersType, FR_CONST) FIFORangeFiltersConfig[4];
} Fr_CCFIFOConfigType;
```

Table 3-104. Structure Fr_CCFIFOConfigType member description

Member	Description
FIFOChannel	Select a receiver FIFO (FIFO A or FIFO B).
FIFOStartIndex	Defines the number of the message buffer header field of the first message buffer of the selected receive FIFO.
FIFODepth	Defines the depth of the selected receive FIFO, i.e. the number of entries.
FIFOEntrySize	Defines the size of the frame data sections for the selected receive FIFO in 2 byte entities.
MessageIDFilterMask	Defines the filter value for the cycle counter filtering.
MessageIDFilterMatch	Defines the filter value for the cycle counter filtering.
FIFORangeFiltersConfig	Defines configuration for four FIFO range filters.

3.6.2.5.4.1 Structure Fr_CCFIFORangeFiltersType

Up to four range filters for each FIFO can be configured. Each filter can be configured either as acceptance or rejection selecting whether FlexRay frames within selected range are received (accepted) into FIFO or not.

Declaration

```
typedef struct
{
  CONST(boolean, FR_CONST) RangeFilterEnable;
  CONST(Fr_CCFIFORangeFilterModeType, FR_CONST) RangeFilterMode;
  CONST(uint16, FR_CONST) RangeFilterLowerInterval;
  CONST(uint16, FR_CONST) RangeFilterUpperInterval;
} Fr_CCFIFORangeFiltersType;
```

Table 3-105. Structure Fr_CCFIFORangeFiltersType member description

Member	Description
RangeFilterEnable	Defines whether relevant range filter is enabled.
RangeFilterMode	Defines the range filter mode (acceptance or rejection).
RangeFilterLowerInterval	Define lower interval boundary.
RangeFilterUpperInterval	Define upper interval boundary.

3.6.2.5.4.2 Example for Fr_CCFIFOConfigType and Fr_CCFifoRangeFiltersType

The FIFO A storage is configured to receive frames on channel A in the ranges of slots from 2 to 3 and from 60 to 69. The payload length is 16 words, FIFO start index is 4 and depth is 3 buffers.

```
/* This structure contains configuration of the receive FIFO for the channel A */
CONST(Fr_CCFifoConfigType, FR_APPL_CONST) FrIfMC0_Clst0_Ctrl0_FIFOA_Cfg_PB =
{
  FR_RECEIVE_FIFOA, /* FIFO receives messages from the channel A */
  4U, /* The first FIFO message buffer */
  3U, /* The number of FIFO message buffers */
  16U, /* Length of the data received into the FIFO */
  0U, /* Message ID filter mask - message ID filtering disabled */
  0U, /* Message ID filter match */
  { /* FIFO A Range filters configuration */
      {TRUE, FR_ACCEPTANCE, 2U, 3U},
      {TRUE, FR_ACCEPTANCE, 60U, 69U},
      {TRUE, FR_REJECTION, 4U, 10U},
      {TRUE, FR_REJECTION, 70U, 72U}
  }
};
```

```
CONST(Fr_CCLPduInfoType, FR_APPL_CONST) FrIfMC0_Clst0_Ctrl0_LPduInfoCfgSet_PB[] =
{
  {FR_TRANSMIT_BUFFER, & FrIfMC0_Clst0_Ctrl0_LPdu0_Cfg_PB, 0U, TRUE, FALSE},
  {FR_TRANSMIT_BUFFER, & FrIfMC0_Clst0_Ctrl0_LPdu1_Cfg_PB, 0U, FALSE, FALSE},
  {FR_RECEIVE_FIFO, & FrIfMC0_Clst0_Ctrl0_FIFOA_Cfg_PB, 255U, TRUE, FALSE},
  {FR_RECEIVE_BUFFER, & FrIfMC0_Clst0_Ctrl0_LPdu3_Cfg_PB, 1U, TRUE, FALSE},
};
```

Example of usage: The FIFO A are configured to receive frames 2, 3 and 60, 61, ...,69.

```
uint8 RxData[32] = {0};
uint8 RxLength = 0;
Fr_RxLPduStatusType RxStatus;
/* The LPdu's with the indexes 0, 2 and 3 are configured */
  Fr_ControllerInit(0);
  Fr_ReceiveRxLPdu(0U, 2U, &RxData[0], &RxStatus, &RxLength);
```

3.6.2.5.5 Structure Fr_CCBufferOffset16Type

Array of this type contains 16-bit offset values for message buffer data areas. These values are used only for message buffer initialization. During transmission/reception operation with message buffer, the similar array of type `Fr_CCBufferAddress32Type` is used.

The address of this array of the type `Fr_CCBufferOffset16Type` should be passed into the relevant BufferInitOffsetTable parameter of the configuration structure of the type `Fr_CCBufferConfigSetType`. Data field offset is used to determine the start address of the corresponding message buffer data field in the FlexRay memory.

The instance of the `Fr_CCBufferOffset16Type` structure is used by the `Fr_ControllerInit()` and `Fr_PrepareLPdu()` functions.

Declaration

```
typedef struct
{
  CONST(uint16, FR_CONST) DataOffset16;
} Fr_CCBufferOffset16Type;
```

Table 3-106. Structure Fr_CCBufferOffset16Type member description

Member	Description
DataOffset16	Offset value of the message buffer data field

3.6.2.5.6 Structure Fr_CCBufferAddress32Type

Array of this type contains the 32-bit addresses (FlexRay memory base plus data area offset) of message buffers data areas. These values are used for message buffer transmission/reception operation.

The address of this array of the type `Fr_CCBufferAddress32Type` should be passed into the relevant BufferOffsetTable parameter of the configuration structure of the type `Fr_CCBufferConfigSetType`.

Data field offset is used to determine the start address of the corresponding message buffer data field in the FlexRay memory according to equation:

DataAddr32 [`Fr_CCBufferAddress32Type`] = DataOffset16 [`Fr_CCBufferOffset16Type`] + CCFlexRayMemoryBaseAddress[`Fr_CCHardwareConfigType`]

The instance of the `Fr_CCBufferAddress32Type` structure is used by the `Fr_TransmitTxLPdu()` and `Fr_ReceiveRxLPdu()` functions.

Declaration

```
typedef struct
{
  CONST(uint32, FR_CONST) DataAddr32;
} Fr_CCBufferAddress32Type;
```

Table 3-107. Structure Fr_CCBufferAddress32Type member description

Member	Description
DataAddr32	Address of the message buffer data field

3.6.2.5.6.1 Example for Fr_CCBufferOffset16Type and Fr_CCBufferAddressType

```
CONST(Fr_CCHardwareConfigType, FR_APPL_CONST) FrMC0_CCHardwareCfgSet_PB =
{
  0xffe5000U, /* Controller base address */
  FR_CC_FLEXRAY_MEMORY_BASE_ADDR, /* FlexRay memory base address */
  FR_CHANNEL_AB, /* FlexRay channels */
  FALSE,  /* FALSE = Dual channel mode selected */
  FR_CLK_SRC_PLL, /* FlexRay protocol engine clock source */
  0U,  /* Bus speed: 10 Mb/s */
  10U, /* System memory access timeout */
  420U, /* SFTOR register value */
  FALSE /* The second absolute timer is disabled */
};
```

```
CONST(Fr_CCBufferOffset16Type, FR_APPL_CONST) FrIfMC0_Clst0_Ctrl0_BOffset16Table_PB[] =
{
  100U, /* 0: Segment 1, Buffer 1 */
  132U, /* 1: Segment 1, Buffer 2 */
  164U, /* 2: Segment 1, Receive Shadow Buffer 1 */
  196U, /* 3: Segment 1, Receive Shadow Buffer 2 */
  228U, /* 4: FIFO A, Buffer 1 */
  260U, /* 5: FIFO A, Buffer 2 */
  292U, /* 6: FIFO A, Buffer 3 */
  324U, /* 7: FIFO B, Buffer 1 */
  356U, /* 8: FIFO B, Buffer 2 */
  388U /* 9: FIFO B, Buffer 3 */
};
```

```
CONST(Fr_CCBufferAddressType, FR_APPL_CONST) FrIfMC0_Clst0_Ctrl0_BAddressTable_PB[] =
{
  FR_CC_FLEXRAY_MEMORY_BASE_ADDR + 100U, /* 0: Segment 1, Buffer 1 */
  FR_CC_FLEXRAY_MEMORY_BASE_ADDR + 132U, /* 1: Segment 1, Buffer 2 */
  FR_CC_FLEXRAY_MEMORY_BASE_ADDR + 164U, /* 2: Segment 1, Receive Shadow Buffer 1 */
  FR_CC_FLEXRAY_MEMORY_BASE_ADDR + 196U, /* 3: Segment 1, Receive Shadow Buffer 2 */
  FR_CC_FLEXRAY_MEMORY_BASE_ADDR + 228U, /* 4: FIFO A, Buffer 1 */
  FR_CC_FLEXRAY_MEMORY_BASE_ADDR + 260U, /* 5: FIFO A, Buffer 2 */
  FR_CC_FLEXRAY_MEMORY_BASE_ADDR + 292U, /* 6: FIFO A, Buffer 3 */
  FR_CC_FLEXRAY_MEMORY_BASE_ADDR + 324U, /* 7: FIFO B, Buffer 1 */
  FR_CC_FLEXRAY_MEMORY_BASE_ADDR + 356U, /* 8: FIFO B, Buffer 2 */
  FR_CC_FLEXRAY_MEMORY_BASE_ADDR + 388U /* 9: FIFO B, Buffer 3 */
};
```

```
CONST(Fr_CCBufferConfigSetType, FR_APPL_CONST) FrIfMC0_Clst0_Ctrl0_BufferCfgSet_PB =
{
  &FrIfMC0_Clst0_Ctrl0_LPduInfoCfgSet_PB[0],  /* LPdu configuration set */
  &FrIfMC0_Clst0_Ctrl0_BAddressTable_PB[0], /* Buffer Addresses Table */
  &FrIfMC0_Clst0_Ctrl0_BOffset16Table_PB[0],  /* Buffer Offsets Table */
  4U, /* Number of items in FrIfMC0_Clst0_Ctrl0_LPduInfoCfgSet_PB */
  16U, /* Data size in buffers segment 1 - gPayloadLengthStatic */
  8U, /* Data size in buffers segment 2 - pPayloadLengthDynMax */
  1U, /* Last MB in segment 1 (Number of MB in Segment1 - 1) */
  1U, /* Last individual MB; (Number of MB in Segment1 + Number of MB in Segment2 - 1) */
  /* Receive shadow buffers configuration */
  2U, /* Ch A, seg 1 - the initial index of the MB header field */
  3U, /* Ch B, seg 1 - the initial index of the MB header field */
  4U, /* Ch A, seg 2 - unused */
  5U /* Ch B, seg 2 - unused */
};
```

Example of usage:

```
Fr_ControllerInit(0);
```

3.7 Implemented Errata Workarounds

Table 3-108. Implemented Errata Workarounds

Name	Headline	Implementation
e8770	Missing TX frames on Channel B when in dual channel mode and Channel A is disabled	This errata is implemented that: • If user configure the single channel mode: use the channel A for both channel A and channel B configurations. • If user configure the dual channel mode: enable both channel A and B. For more detail, please read carefully on the errata document.[Reference List]

Chapter 4
Tresos Configuration Plug-in

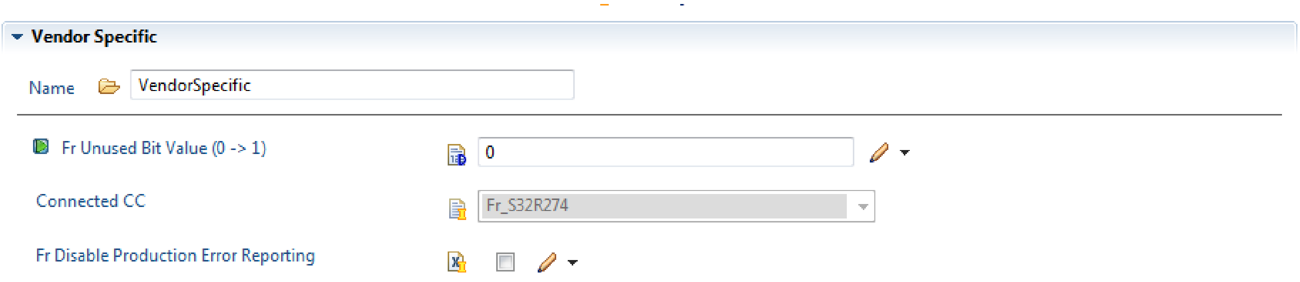
This chapter describes the Tresos configuration plug-in for the FR Driver. The most of the parameters are described below.

4.1 FrGeneral

Parameters are described in the AUTOSAR FlexRay Driver software specification document [Table 2-3]. The FlexRay driver supports one CC, the `FrNumCtrlSupported` parameter can be set at most to value 1, because there is only one CC available on the NXP Semiconductor MCUs

4.1.1 FrGeneral - VendorSpecific

FrGeneral container contains vendor specific configuration parameters which are relevant for all multiple configurations.



Vendor Specific

Name: VendorSpecific

Fr Unused Bit Value (0 -> 1): 0

Connected CC: Fr_S32R274

Fr Disable Production Error Reporting: ☐

Figure 4-1. FrGeneral Container - VendorSpecific Section

Table 4-1. FrUnusedBitValue

Description	Specifies that the function Fr_TransmitTxLPdu() sets the remaining bits in the CC's transmission resource to the configured value given by parameter FrUnusedBitValue. This feature is reused from Autosar FlexRay Driver specification v2.1 REV0019.
Class	Implementation Specific Parameter

Table continues on the next page...

Table 4-1. FrUnusedBitValue (continued)

Range	True, False
Default	Disabled
Source File	Fr_Cfg.h
Source Representation	#define FR_UNUSED_BIT_VALUE 0xFFU #define FR_UNUSED_BIT_VALUE_16 0xFFFFU #define FR_UNUSED_BIT_VALUE_32 0xFFFFFFFFU

Table 4-2. ConnectedCC

Description	Informs about the type of used NXP Semiconductor FlexRay device.
Class	Implementation Specific Parameter
Range	FR_MPC574XG
Default	FR_MPC574XG
Source File	none
Source Representation	none

Table 4-3. FrDisableDemReportErrorStatus

Description	Globally disables (True) or enables (False) DEM errors/events reporting.
Class	Implementation Specific Parameter
Range	True, False
Default	False
Source File	Fr_PBcfg_[VariantName].c

Source Representation	STATIC CONST(Mcal_DemErrorType, FR_APPL_CONST)
	FrMC0_DemEventParameter_PB =
	{
	  (VAR(uint32, AUTOMATIC))STD_ON, /* DemEventParameter_0: DEM reporting is enabled */
	  DemEventParameter_0 /* DEM ID */
	};
	
	CONST(Fr_ConfigurationType, FR_APPL_CONST) Fr_Config_0_VS0 =
	{
	  &FrMC0_CCHardwareCfgSet_PB,
	  &FrIfMC0_Clst0_Ctrl0_BufferCfgSet_PB,
	  &FrMC0_CCLowLevelCfgSet_PB,
	  &FrMC0_DemEventParameter_PB
	};

Table 4-4. FrEnableUserModeSupport

Description	Disable or enable user mode sopporting.
Class	Implementation Specific Parameter
Range	True, False

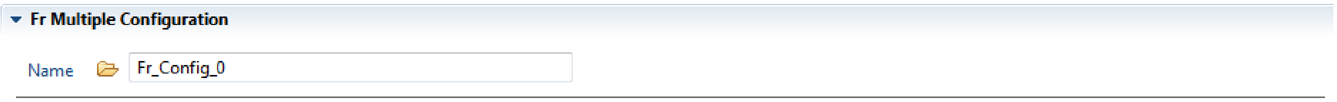
Table continues on the next page...

Table 4-4. FrEnableUserModeSupport (continued)

Default	False
Source File	none
Source Representation	none

4.2 FrMultipleConfiguration - FrController

FrMultipleConfiguration container configuration for FlexRay CC.



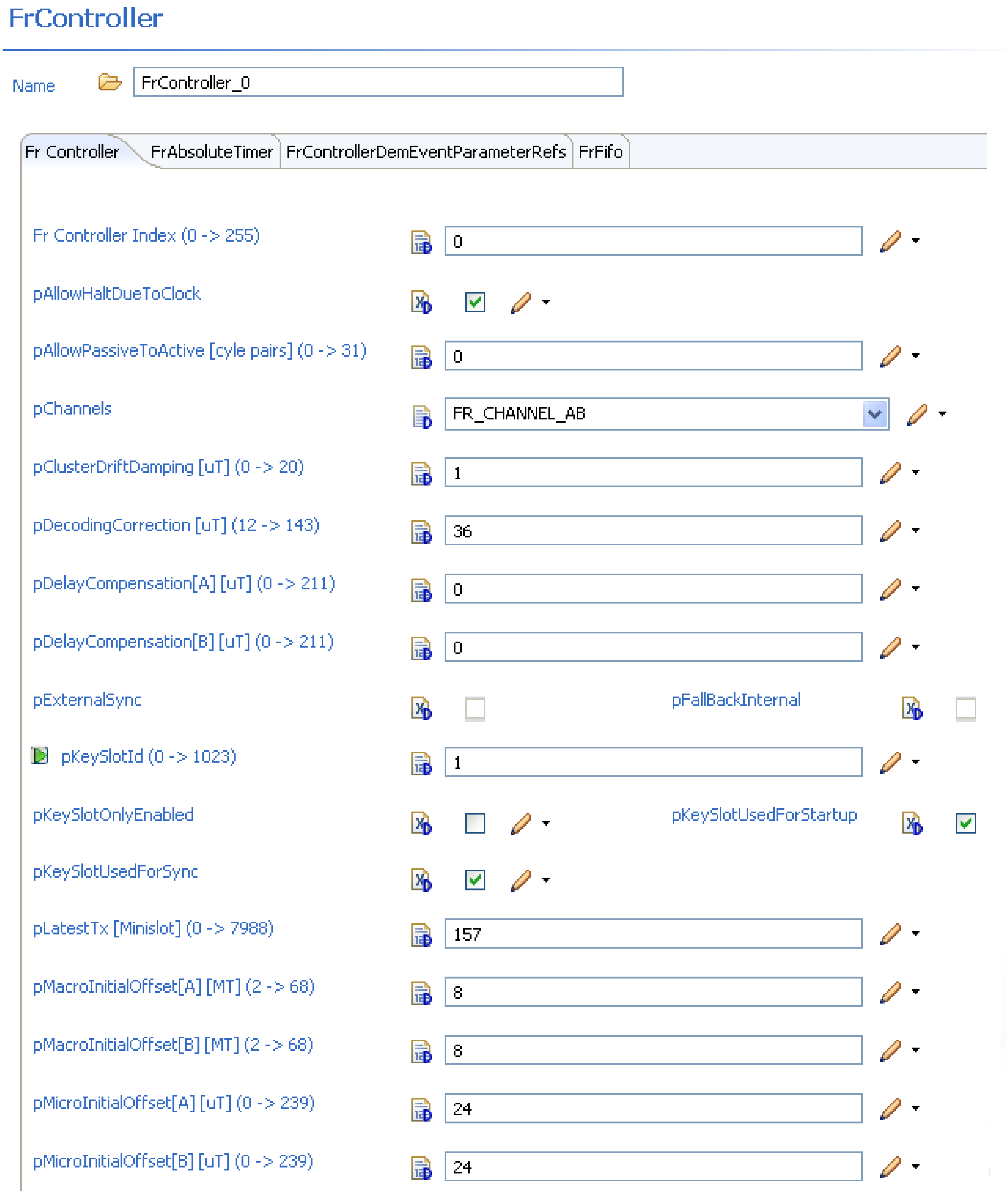
Fr Multiple Configuration

Name: Fr_Config_0

Figure 4-2. FrMultipleConfiguration Container

4.2.1 FrController - General

FrController container contains the configuration data for FlexRay CC controlled by one FlexRay driver. The container is divided into several sections which are described below.



FrController

Name: FrController_0

Fr Controller Index (0 -> 255): 0

pAllowHaltDueToClock: ☑

pAllowPassiveToActive [cyle pairs] (0 -> 31): 0

pChannels: FR_CHANNEL_AB

pClusterDriftDamping [uT] (0 -> 20): 1

pDecodingCorrection [uT] (12 -> 143): 36

pDelayCompensation[A] [uT] (0 -> 211): 0

pDelayCompensation[B] [uT] (0 -> 211): 0

pExternalSync: □ pFallBackInternal: □

pKeySlotId (0 -> 1023): 1

pKeySlotOnlyEnabled: □ pKeySlotUsedForStartup: ☑

pKeySlotUsedForSync: ☑

pLatestTx [Minislot] (0 -> 7988): 157

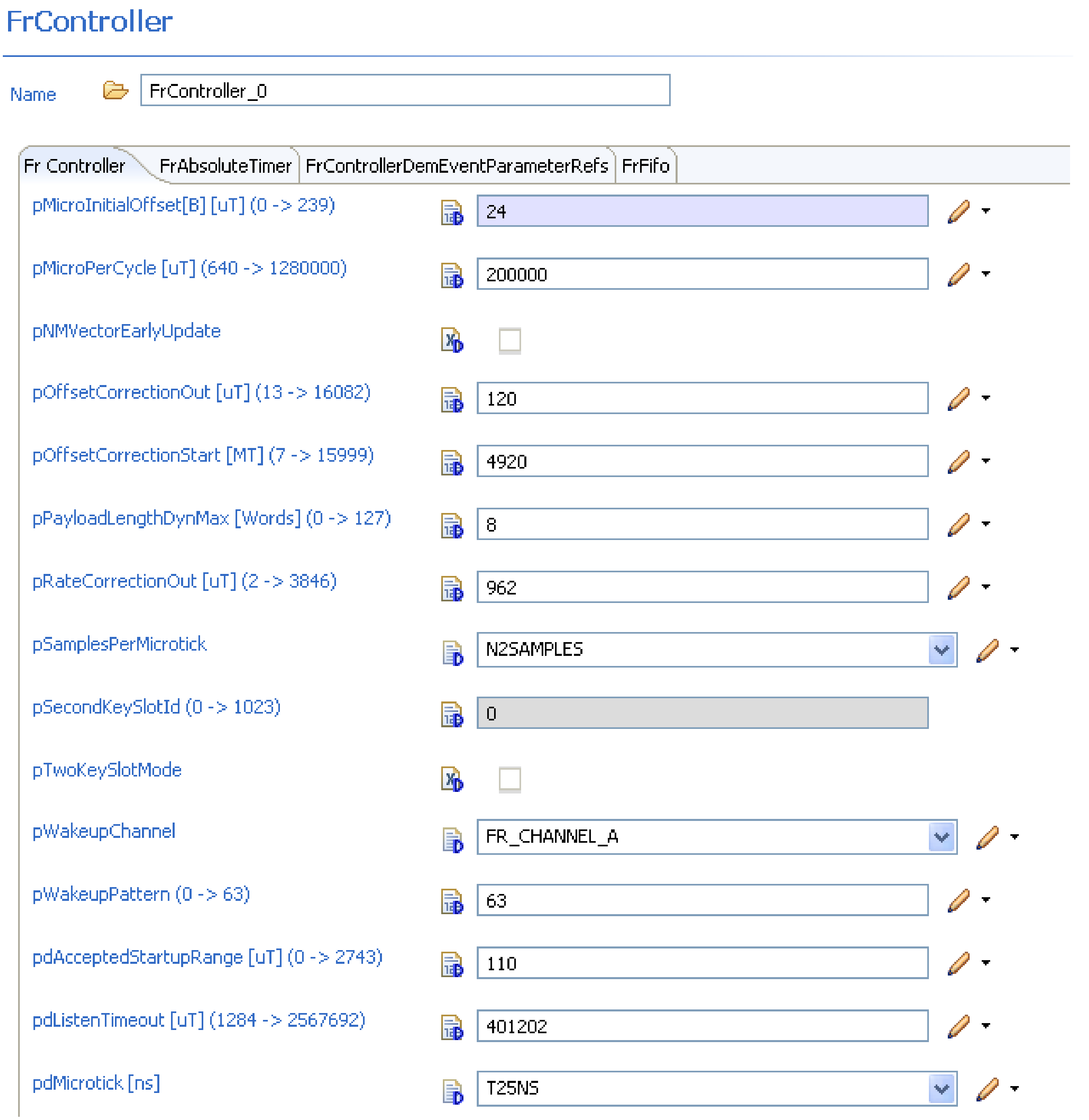
pMacroInitialOffset[A] [MT] (2 -> 68): 8

pMacroInitialOffset[B] [MT] (2 -> 68): 8

pMicroInitialOffset[A] [uT] (0 -> 239): 24

pMicroInitialOffset[B] [uT] (0 -> 239): 24

Figure 4-3. FrController Container - General Section Part 1



FrController

Name FrController_0

pMicroInitialOffset[B] [uT] (0 -> 239) 24

pMicroPerCycle [uT] (640 -> 1280000) 200000

pNMVectorEarlyUpdate □

pOffsetCorrectionOut [uT] (13 -> 16082) 120

pOffsetCorrectionStart [MT] (7 -> 15999) 4920

pPayloadLengthDynMax [Words] (0 -> 127) 8

pRateCorrectionOut [uT] (2 -> 3846) 962

pSamplesPerMicrotick N2SAMPLES

pSecondKeySlotId (0 -> 1023) 0

pTwoKeySlotMode □

pWakeupChannel FR_CHANNEL_A

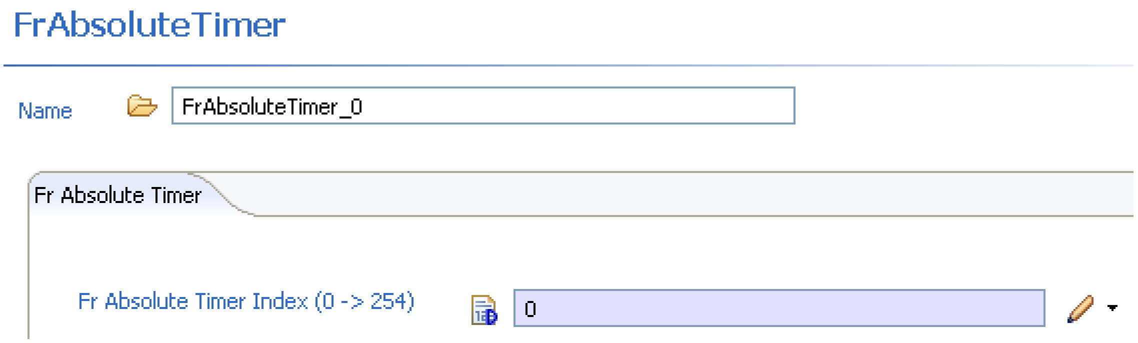
pWakeupPattern (0 -> 63) 63

pdAcceptedStartupRange [uT] (0 -> 2743) 110

pdListenTimeout [uT] (1284 -> 2567692) 401202

pdMicrotick [ns] T25NS

Figure 4-4. FrController Container - General Section Part 2



FrAbsoluteTimer

Name: FrAbsoluteTimer_0

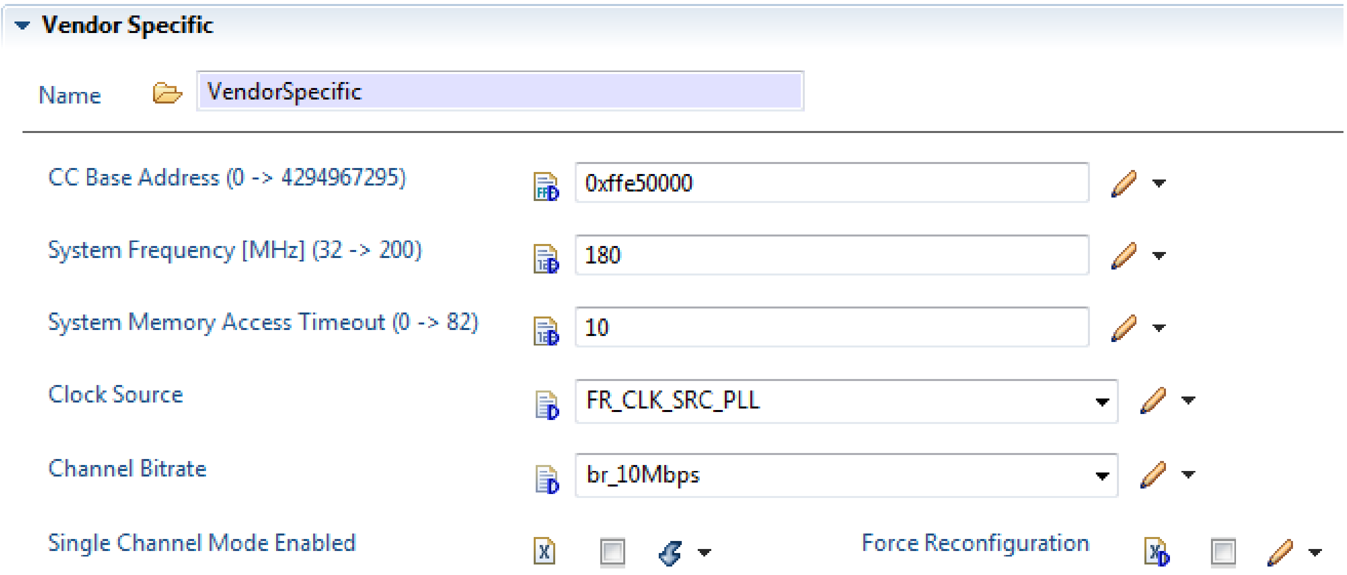
Fr Absolute Timer

Fr Absolute Timer Index (0 -> 254): 0

Figure 4-5. FrController Container - FrAbsoluteTimer Container

All general configuration data relevant for given CC should be configured in the General section of the FrController container. These data are stored into an instance of the type `Fr_CCLowLevelConfigSetType` in the Fr_PBcfg_[VariantName].c by the Tresos generator tool. For detailed description see the AUTOSAR FlexRay Driver software specification document [Table 2-3].

4.2.2 FrController - VendorSpecific



Vendor Specific

Name: VendorSpecific

CC Base Address (0 -> 4294967295): 0xffe50000

System Frequency [MHz] (32 -> 200): 180

System Memory Access Timeout (0 -> 82): 10

Clock Source: FR_CLK_SRC_PLL

Channel Bitrate: br_10Mbps

Single Channel Mode Enabled: □

Force Reconfiguration: □

Figure 4-6. FrController Container - VendorSpecific Section

All vendor specific configuration data relevant for given CC should be configured in the VendorSpecific section of the FrController container. These data are stored into instance of the type `Fr_CCHardwareConfigType` in the Fr_PBcfg_[VariantName].c file by the Tresos generator tool. See sections Structure Fr_CCHardwareConfigType for more information.

Table 4-5. CCBaseAddress

Description	Configures the base address of the FlexRay module. This value depends on the FlexRay module used.
Class	Implementation Specific Parameter
Range	0x0 to 0xFFFFFFFF
Default	0xffe5000U
Source File	Fr_PBcfg_[VariantName].c

Source Representation	`CONST(Fr_CCHardwareConfigType, FR_APPL_CONST) \`
	`FrMC0_CCHardwareCfgSet_PB =`
	`{`
	`  0xffe5000U,  /* Controller base address */`
	`  FR_CC_FLEXRAY_MEMORY_BASE_ADDR,  /* FlexRay memory base address */`
	`  FR_CHANNEL_AB,  /* FlexRay channels */`
	`  FALSE,  /* FALSE = Dual channel mode selected */`
	`  FR_CLK_SRC_PLL,  /* FlexRay protocol engine clock source */`
	`  0U,  /* Bus speed: 10 Mb/s */`
	`  10U,  /* System memory access timeout */`
	`  476U,  /* SFTOR register value */`
	`  FALSE  /* The second absolute timer is disabled */`
	`};`

Table 4-6. SystemFrequency and SystemMemoryAccessTimeOut

Description	The Timeout value corresponds directly to a certain acceptable number of wait states on the system bus. To ensure reliable operation of the FlexRay CC, the Timeout value in the System Memory Access Time-Out register and the CHI clock frequency f_{CHI} (SystemFrequency) in [MHz] must fulfill the following equation. $0<SYMATOR[Timeout]\times0.45<f_{CHI}-8$
Class	Implementation Specific Parameter
Range	0 to 82
Default	10U
Source File	Fr_PBcfg_[VariantName].c

Source Representation	`CONST(Fr_CCHardwareConfigType, FR_APPL_CONST) \`
	`FrMC0_CCHardwareCfgSet_PB =`
	`{`
	`  0xffe5000U,  /* Controller base address */`
	`  FR_CC_FLEXRAY_MEMORY_BASE_ADDR,  /* FlexRay memory base address */`
	`  FR_CHANNEL_AB,  /* FlexRay channels */`
	`  FALSE,  /* FALSE = Dual channel mode selected */`
	`  FR_CLK_SRC_PLL,  /* FlexRay protocol engine clock source */`
	`  0U,  /* Bus speed: 10 Mb/s */`
	`  10U,  /* System memory access timeout */`
	`  476U,  /* SFTOR register value */`
	`  FALSE  /* The second absolute timer is disabled */`
	`};`

Table 4-7. ClockSource

Description	Allows the user to select the proper protocol engine clock source. The value `FR_CLK_SRC_PLL` represents the PE clock source generated by on-chip PLL, value `FR_CLK_SRC_CRYSTAL_OSCILLATOR` represents the PE clock source generated by on-chip crystal oscillator.
Class	Implementation Specific Parameter
Range	`FR_CLK_SRC_PLL`, `FR_CLK_SRC_CRYSTAL_OSCILLATOR`
Default	`FR_CLK_SRC_PLL`
Source File	Fr_PBcfg_[VariantName].c

Source Representation	CONST(Fr_CCHardwareConfigType, FR_APPL_CONST) \
	FrMC0_CCHardwareCfgSet_PB =
	{
	  0xffe5000U, /* Controller base address */
	  FR_CC_FLEXRAY_MEMORY_BASE_ADDR, /* FlexRay memory base address */
	  FR_CHANNEL_AB, /* FlexRay channels */
	  FALSE, /* FALSE = Dual channel mode selected */
	  FR_CLK_SRC_PLL, /* FlexRay protocol engine clock source */
	  0U, /* Bus speed: 10 Mb/s */
	  10U, /* System memory access timeout */
	  476U, /* SFTOR register value */
	  FALSE /* The second absolute timer is disabled */
	};

Table 4-8. ChannelBitrate

Description	Allows the user to select the required FlexRay communication bit rate.
Class	Implementation Specific Parameter
Range	br_10Mbps, br_8Mbps, br_5Mbps, br_2500kbps
Default	br_10Mbps
Source File	Fr_PBcfg_[VariantName].c

Source Representation	CONST(Fr_CCHardwareConfigType, FR_APPL_CONST) \
	FrMC0_CCHardwareCfgSet_PB =
	{
	  0xffe5000U, /* Controller base address */
	  FR_CC_FLEXRAY_MEMORY_BASE_ADDR, /* FlexRay memory base address */
	  FR_CHANNEL_AB, /* FlexRay channels */
	  FALSE, /* FALSE = Dual channel mode selected */
	  FR_CLK_SRC_PLL, /* FlexRay protocol engine clock source */
	  0U, /* Bus speed: 10 Mb/s */
	  10U, /* System memory access timeout */
	  476U, /* SFTOR register value */
	  FALSE /* The second absolute timer is disabled */
	};

Table 4-9. SingleChannelModeEnabled

Description	Enables the single channel mode, otherwise the dual channel mode is configured.
Class	Implementation Specific Parameter

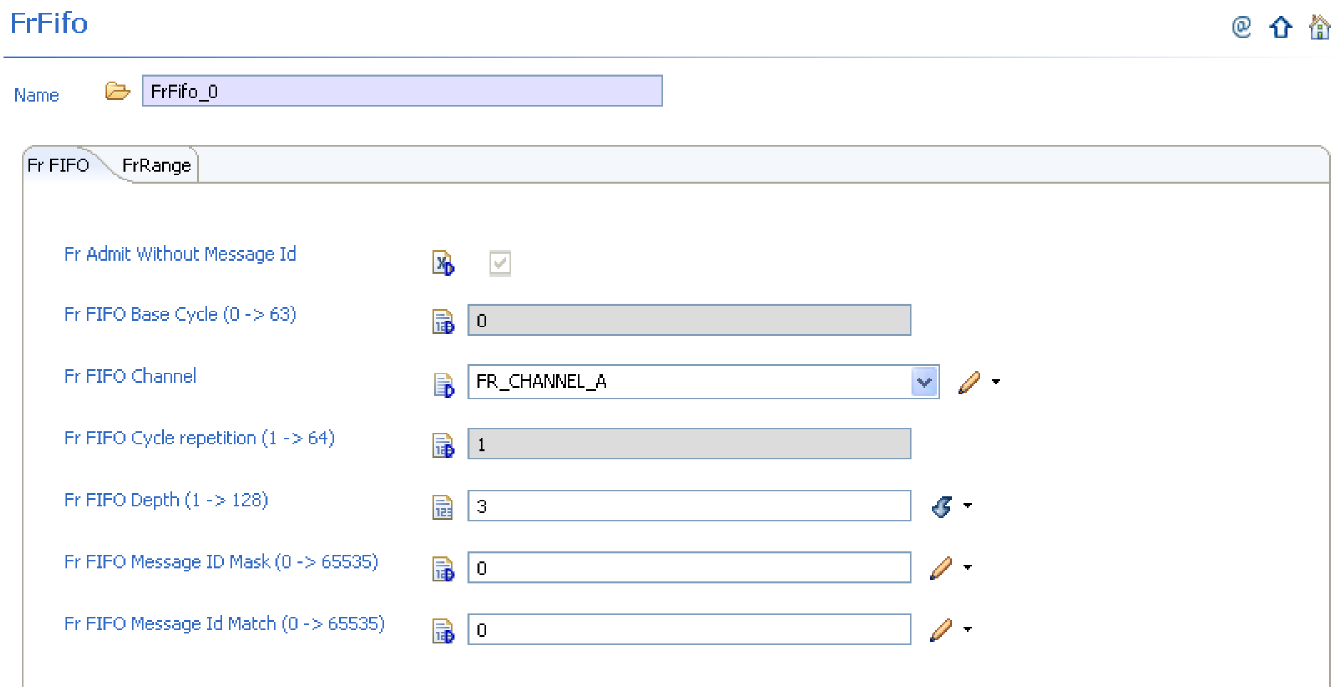
Table continues on the next page...

Table 4-9. SingleChannelModeEnabled (continued)

Range	True, False
Default	False
Source File	Fr_PBcfg_[VariantName].c

Source Representation	`CONST(Fr_CCHardwareConfigType, FR_APPL_CONST) \`
	`FrMC0_CCHardwareCfgSet_PB =`
	`{`
	`  0xffe5000U,  /* Controller base address */`
	`  FR_CC_FLEXRAY_MEMORY_BASE_ADDR,  /* FlexRay memory base address */`
	`  FR_CHANNEL_AB,  /* FlexRay channels */`
	`  **FALSE**,  /* FALSE = Dual channel mode selected */`
	`  FR_CLK_SRC_PLL,  /* FlexRay protocol engine clock source */`
	`  0U,  /* Bus speed: 10 Mb/s */`
	`  10U,  /* System memory access timeout */`
	`  476U,  /* SFTOR register value */`
	`  FALSE  /* The second absolute timer is disabled */`
	`};`

4.2.3 FrController - FrFifo



FrFifo

Name FrFifo_0

Fr FIFO | FrRange

Fr Admit Without Message Id ☑

Fr FIFO Base Cycle (0 -> 63) 0

Fr FIFO Channel FR_CHANNEL_A

Fr FIFO Cycle repetition (1 -> 64) 1

Fr FIFO Depth (1 -> 128) 3

Fr FIFO Message ID Mask (0 -> 65535) 0

Fr FIFO Message Id Match (0 -> 65535) 0

Figure 4-7. FrController Container - FrFifo Section

Fifo parameters are described in the AUTOSAR FlexRay Driver software specification document [Table 2-3]. These data are stored into instance of the types `Fr_CCFIFOConfigType` in the Fr_PBcfg_[VariantName].c file by the Tresos generator tool. See sections Structure Fr_CCFIFOConfigType and Structure Fr_CCFIFORangeFiltersType for more information.

Table 4-10. FrFifoChannel

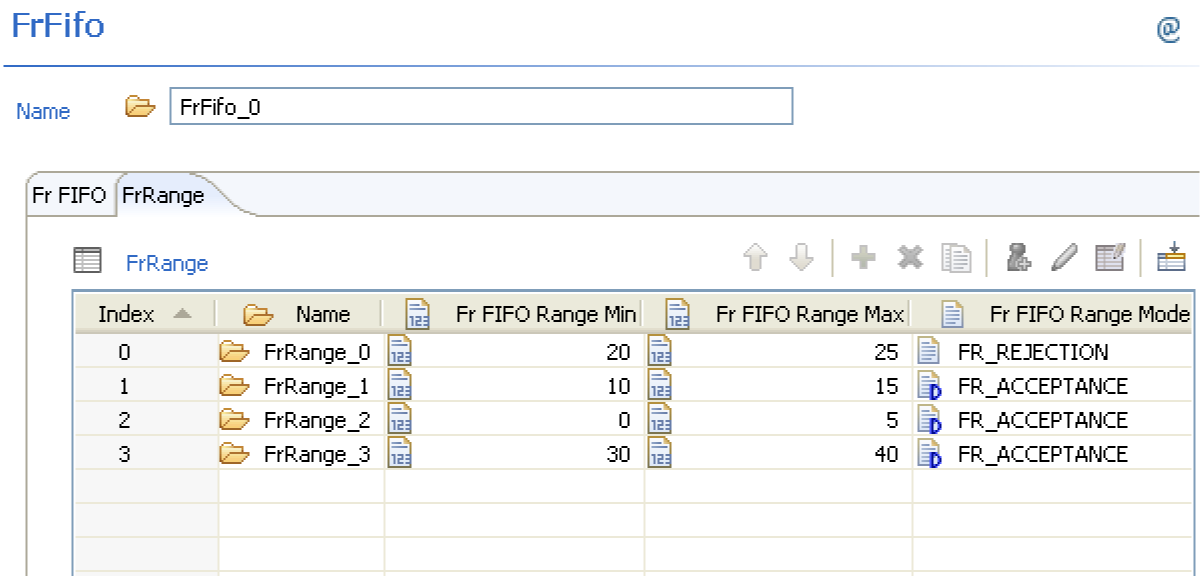
Description	Defines the channel for FIFO reception.
Class	Implementation Specific Parameter
Range	FR_CHANNEL_A, FR_CHANNEL_B
Default	FR_CHANNEL_A
Source File	Fr_PBcfg_[VariantName].c

Source Representation	CONST(Fr_CCFIFOConfigType, FR_APPL_CONST) \
	FrIfMC0_Clst0_Ctrl0_FIFOA_Cfg_PB =
	{
	  FR_RECEIVE_FIFOA, /* FIFO receives messages from the channel A */
	  12U, /* The first FIFO message buffer */
	  1U, /* The number of FIFO message buffers */
	  8U, /* Length of the data received into the FIFO */
	  0U, /* Message ID filter mask - message ID filtering disabled */
	  0U, /* Message ID filter match */
	  {
	      {TRUE, FR_ACCEPTANCE, 10U, 15U},
	      {FALSE, FR_ACCEPTANCE, 0U, 0U},
	      {FALSE, FR_ACCEPTANCE, 0U, 0U},
	      {FALSE, FR_ACCEPTANCE, 0U, 0U},
	  }
	};

Table 4-11. FrFifoDepth

Description	Defines the depth of the receive FIFO, i.e. the number of entries.
Class	Implementation Specific Parameter
Range	1 to 128
Default	32
Source File	Fr_PBcfg_[VariantName].c

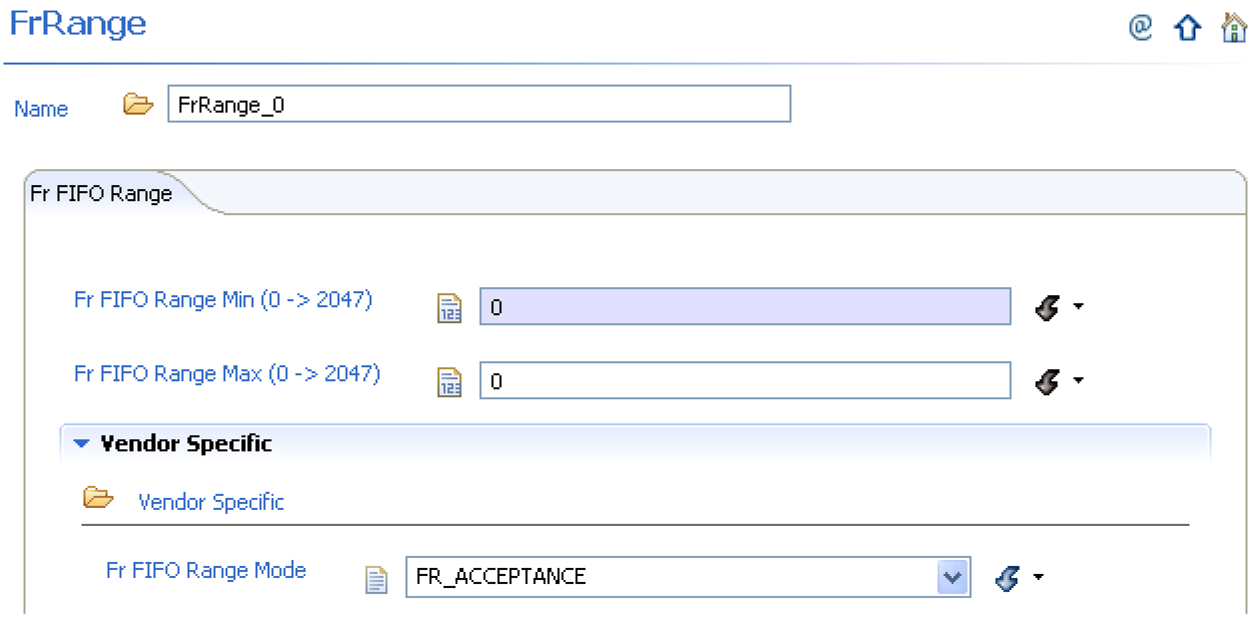
Source Representation	CONST(Fr_CCFIFOConfigType, FR_APPL_CONST) \
	FrIfMC0_Clst0_Ctrl0_FIFOA_Cfg_PB =
	{
	  FR_RECEIVE_FIFOA, /* FIFO receives messages from the channel A */
	  12U, /* The first FIFO message buffer */
	  1U, /* The number of FIFO message buffers */
	  8U, /* Length of the data received into the FIFO */
	  0U, /* Message ID filter mask - message ID filtering disabled */
	  0U, /* Message ID filter match */
	  {
	      {TRUE, FR_ACCEPTANCE, 10U, 15U},
	      {FALSE, FR_ACCEPTANCE, 0U, 0U},
	      {FALSE, FR_ACCEPTANCE, 0U, 0U},
	      {FALSE, FR_ACCEPTANCE, 0U, 0U},
	    }
	};



Name: FrFifo_0

Index	Name	Fr FIFO Range Min	Fr FIFO Range Max	Fr FIFO Range Mode
0	FrRange_0	20	25	FR_REJECTION
1	FrRange_1	10	15	FR_ACCEPTANCE
2	FrRange_2	0	5	FR_ACCEPTANCE
3	FrRange_3	30	40	FR_ACCEPTANCE

Figure 4-8. FrController Container - FrFifo Section - the List of the FrRange Subcontainers



Name: FrRange_0

Fr FIFO Range Min (0 -> 2047): 0

Fr FIFO Range Max (0 -> 2047): 0

Fr FIFO Range Mode: FR_ACCEPTANCE

Figure 4-9. FrController Container - FrFifo Section - One Instance of FrRange Subcontainer

Table 4-12. FrFifoRangeMin

Description	Defines range filter value of lower interval boundary.
Class	Implementation Specific Parameter
Range	0 to 2047
Default	n/a
Source File	Fr_PBcfg_[VariantName].c

Table continues on the next page...

Table 4-12. FrFifoRangeMin (continued)

<table>
<tr><td><b>Source Representation</b></td><td><pre>
CONST(Fr_CCFIFOConfigType, FR_APPL_CONST) \
FrIfMC0_Clst0_Ctrl0_FIFOA_Cfg_PB =
{
  FR_RECEIVE_FIFOA, /* FIFO receives messages from the channel A */
  12U, /* The first FIFO message buffer */
  1U, /* The number of FIFO message buffers */
  8U, /* Length of the data received into the FIFO */
  0U, /* Message ID filter mask - message ID filtering disabled */
  0U, /* Message ID filter match */
  {
      {TRUE, FR_ACCEPTANCE, 10U, 15U},
      {FALSE, FR_ACCEPTANCE, 0U, 0U},
      {FALSE, FR_ACCEPTANCE, 0U, 0U},
      {FALSE, FR_ACCEPTANCE, 0U, 0U},
  }
};
</pre></td></tr>
</table>

Table 4-13. FrFifoRangeMax

<table>
<tr><td><b>Description</b></td><td>Defines range filter value of upper interval boundary.</td></tr>
<tr><td><b>Class</b></td><td>Implementation Specific Parameter</td></tr>
<tr><td><b>Range</b></td><td>0 to 2047</td></tr>
<tr><td><b>Default</b></td><td>n/a</td></tr>
<tr><td><b>Source File</b></td><td>Fr_PBcfg_[VariantName].c</td></tr>
<tr><td><b>Source Representation</b></td><td><pre>
CONST(Fr_CCFIFOConfigType, FR_APPL_CONST) \
FrIfMC0_Clst0_Ctrl0_FIFOA_Cfg_PB =
{
  FR_RECEIVE_FIFOA, /* FIFO receives messages from the channel A */
  12U, /* The first FIFO message buffer */
  1U, /* The number of FIFO message buffers */
  8U, /* Length of the data received into the FIFO */
  0U, /* Message ID filter mask - message ID filtering disabled */
  0U, /* Message ID filter match */
  {
      {TRUE, FR_ACCEPTANCE, 10U, 15U},
      {FALSE, FR_ACCEPTANCE, 0U, 0U},
      {FALSE, FR_ACCEPTANCE, 0U, 0U},
      {FALSE, FR_ACCEPTANCE, 0U, 0U},
  }
};
</pre></td></tr>
</table>

Table 4-14. FrFifoARangeMode

<table>
<tr><td><b>Description</b></td><td>Defines range filter mode, i.e. if range filter is configured as acceptance or rejection filter.</td></tr>
<tr><td><b>Class</b></td><td>Implementation Specific Parameter</td></tr>
<tr><td><b>Range</b></td><td>Acceptance, Rejection</td></tr>
<tr><td><b>Default</b></td><td>Acceptance</td></tr>
<tr><td><b>Source File</b></td><td>Fr_PBcfg_[VariantName].c</td></tr>
<tr><td><b>Source Representation</b></td><td><pre>
{
  FR_RECEIVE_FIFOA, /* FIFO receives messages from the channel A */
</pre></td></tr>
</table>

Table 4-14. FrFifoARangeMode

```
  12U, /* The first FIFO message buffer */
  1U, /* The number of FIFO message buffers */
  8U, /* Length of the data received into the FIFO */
  0U, /* Message ID filter mask - message ID filtering disabled */
  0U, /* Message ID filter match */
  {
      {TRUE, FR_ACCEPTANCE, 10U, 15U},
      {FALSE, FR_ACCEPTANCE, 0U, 0U},
      {FALSE, FR_ACCEPTANCE, 0U, 0U},
      {FALSE, FR_ACCEPTANCE, 0U, 0U},
    }
};
```

Note

At least one LPdu needs to be assigned for one slot from range for which FIFO is configured.

4.2.3.1 Generation of FIFO Configuration

During generation of FIFO configuration, FIFO range filters and the list of virtual buffers in FrIfCluster container are analysed. The FIFO configuration is generated for each virtual buffer configured for reception on given channel and assigned to a frame which shall be received by FIFO storage. Then these virtual buffers are not configured as standard receive message buffer. Basically, it is possible to assign more virtual buffers for one FIFO storage and then obtain received data by calling of the `Fr_ReceiveRxLPdu()` function with `Fr_LPduIdx` index equal to one of assigned FIFO virtual buffers; the `Fr_ReceiveRxLPdu()` function will operate with FIFO storage instead of standard receive message buffer.

Please, see FIFO Operations section for more information.

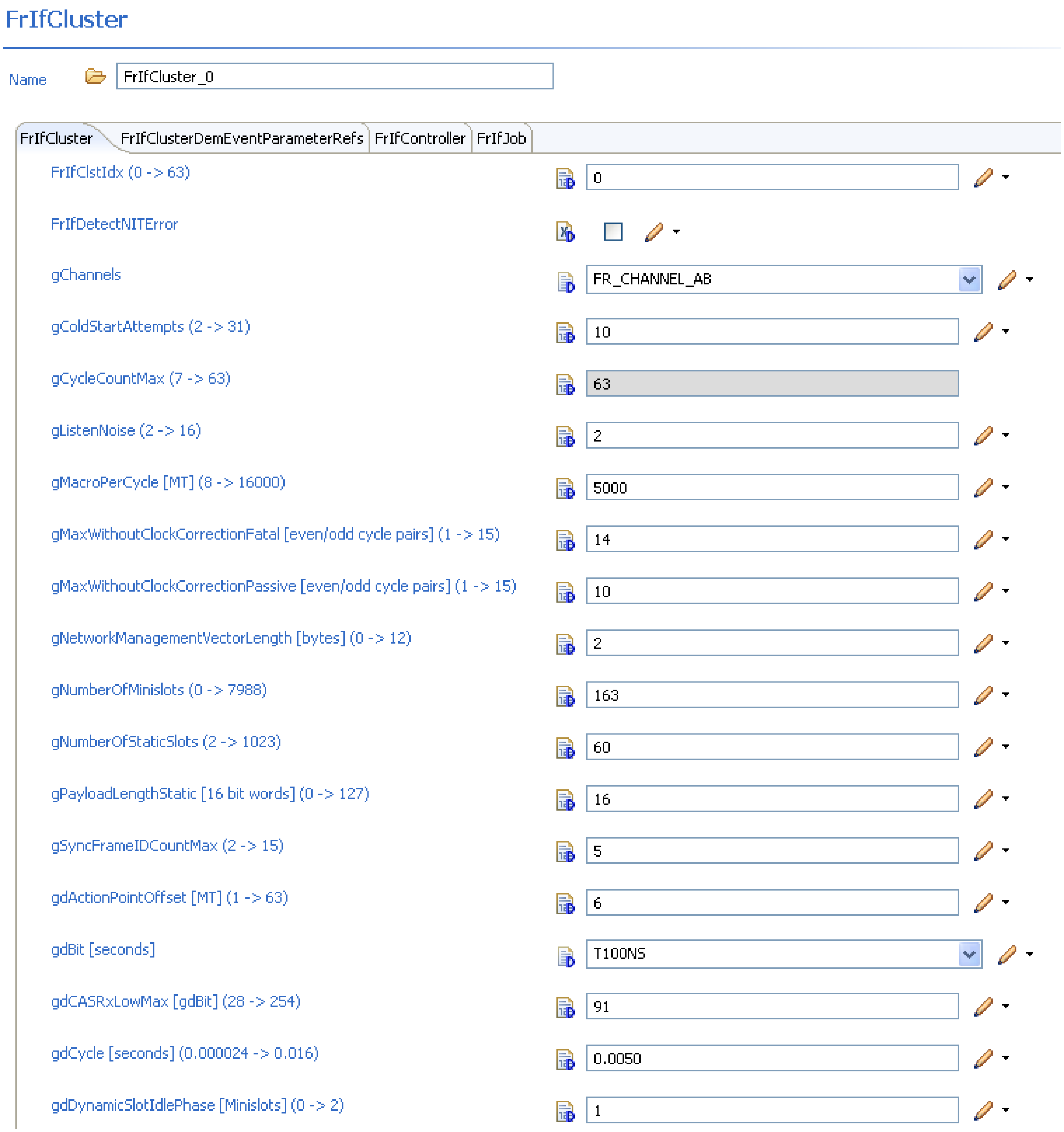
4.3 FrIfConfig - FrIfCluster

Cluster and the LPdu configuration parameters are stored in the FlexRay Interface variants.

FrIfConfig Variant contains all multiple variant configurations for FlexRay Interface.

Note

For each FlexRay Interface multiple variant configuration set has to exist relevant FlexRay multiple variant configuration set, i.e. for each FrIfCluster sub-container has to exist one FrMultipleConfiguration sub-container.

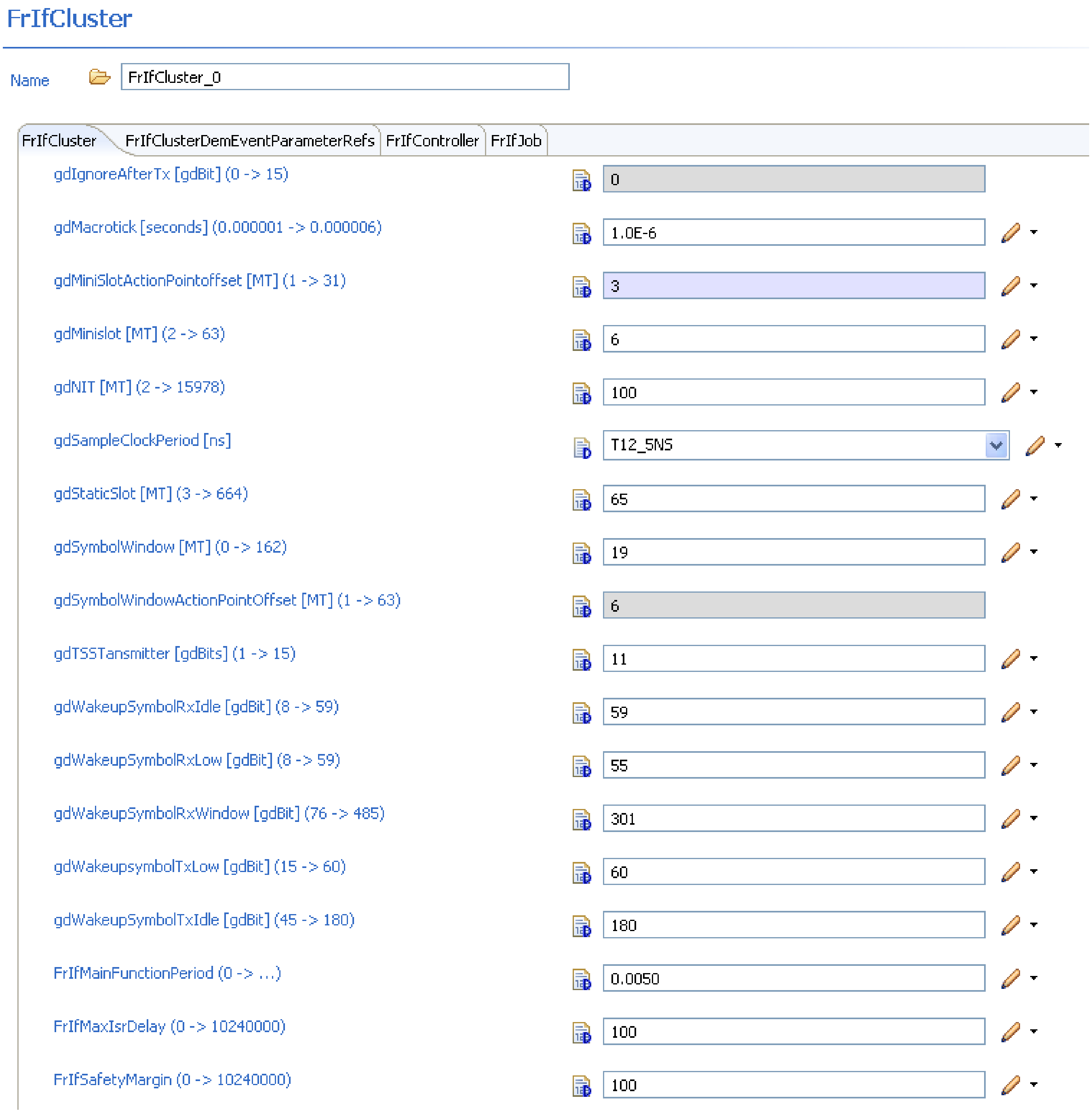


FrIfCluster

Name: FrIfCluster_0

FrIfClstIdx (0 -> 63)	0
FrIfDetectNITError	□
gChannels	FR_CHANNEL_AB
gColdStartAttempts (2 -> 31)	10
gCycleCountMax (7 -> 63)	63
gListenNoise (2 -> 16)	2
gMacroPerCycle [MT] (8 -> 16000)	5000
gMaxWithoutClockCorrectionFatal [even/odd cycle pairs] (1 -> 15)	14
gMaxWithoutClockCorrectionPassive [even/odd cycle pairs] (1 -> 15)	10
gNetworkManagementVectorLength [bytes] (0 -> 12)	2
gNumberOfMinislots (0 -> 7988)	163
gNumberOfStaticSlots (2 -> 1023)	60
gPayloadLengthStatic [16 bit words] (0 -> 127)	16
gSyncFrameIDCountMax (2 -> 15)	5
gdActionPointOffset [MT] (1 -> 63)	6
gdBit [seconds]	T100NS
gdCASRxLowMax [gdBit] (28 -> 254)	91
gdCycle [seconds] (0.000024 -> 0.016)	0.0050
gdDynamicSlotIdlePhase [Minislots] (0 -> 2)	1

Figure 4-10. FrIfCluster Container - General Section Part 1



FrIfCluster

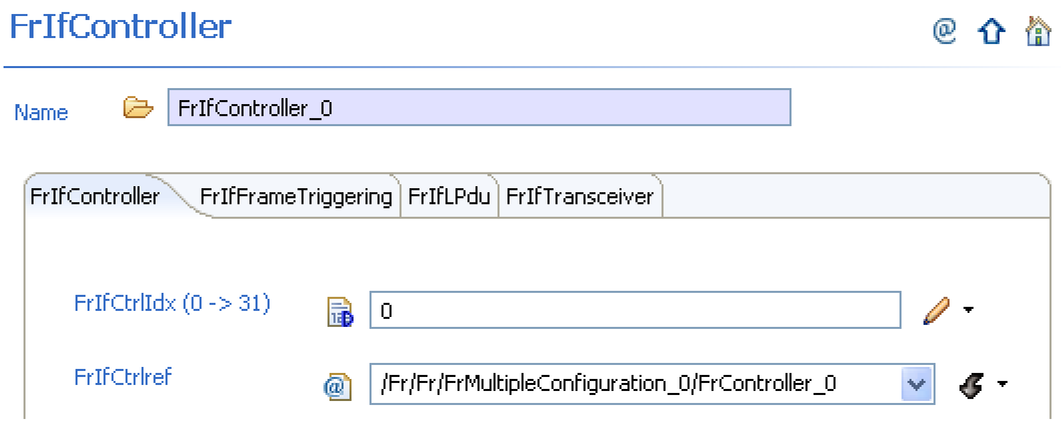
Name FrIfCluster_0

gdIgnoreAfterTx [gdBit] (0 -> 15)	0
gdMacrotick [seconds] (0.000001 -> 0.000006)	1.0E-6
gdMiniSlotActionPointoffset [MT] (1 -> 31)	3
gdMinislot [MT] (2 -> 63)	6
gdNIT [MT] (2 -> 15978)	100
gdSampleClockPeriod [ns]	T12_5NS
gdStaticSlot [MT] (3 -> 664)	65
gdSymbolWindow [MT] (0 -> 162)	19
gdSymbolWindowActionPointOffset [MT] (1 -> 63)	6
gdTSSTansmitter [gdBits] (1 -> 15)	11
gdWakeupSymbolRxIdle [gdBit] (8 -> 59)	59
gdWakeupSymbolRxLow [gdBit] (8 -> 59)	55
gdWakeupSymbolRxWindow [gdBit] (76 -> 485)	301
gdWakeupsymbolTxLow [gdBit] (15 -> 60)	60
gdWakeupSymbolTxIdle [gdBit] (45 -> 180)	180
FrIfMainFunctionPeriod (0 -> ...)	0.0050
FrIfMaxIsrDelay (0 -> 10240000)	100
FrIfSafetyMargin (0 -> 10240000)	100

Figure 4-11. FrIfCluster Container - General Section Part 2

All general cluster configuration data relevant for given cluster should be configured in the General section of the FrIfCluster container. These data are stored into an instance of the type `Fr_CCLowLevelConfigSetType` in the Fr_PBcfg_[VariantName].c file by the Tresos generator tool.

4.3.1 FrIfCluster - FrIfController



FrIfController

Name: FrIfController_0

FrIfCtrlIdx (0 -> 31): 0

FrIfCtrlref: /Fr/Fr/FrMultipleConfiguration_0/FrController_0

Figure 4-12. FrIfCluster Container - FrIfController Section

The LPdu specific configuration data relevant for given CC should be configured in the FrIfController section of the FrIfCluster container. These data are stored into instances of the types `Fr_CCBufferConfigSetType`, `Fr_CCLPduInfoType`, `Fr_CCTransmitBufferConfigType`, `Fr_CCReceiveBufferConfigType`, `Fr_CCBufferOffset16Type` and `Fr_CCBufferAddress32Type` in the Fr_PBcfg_[VariantName].c file by the Tresos generator tool. See chapter Structure Fr_CCBufferConfigSetType and all relevant chapters for more information.

Table 4-15. FrIfCtrlIdx

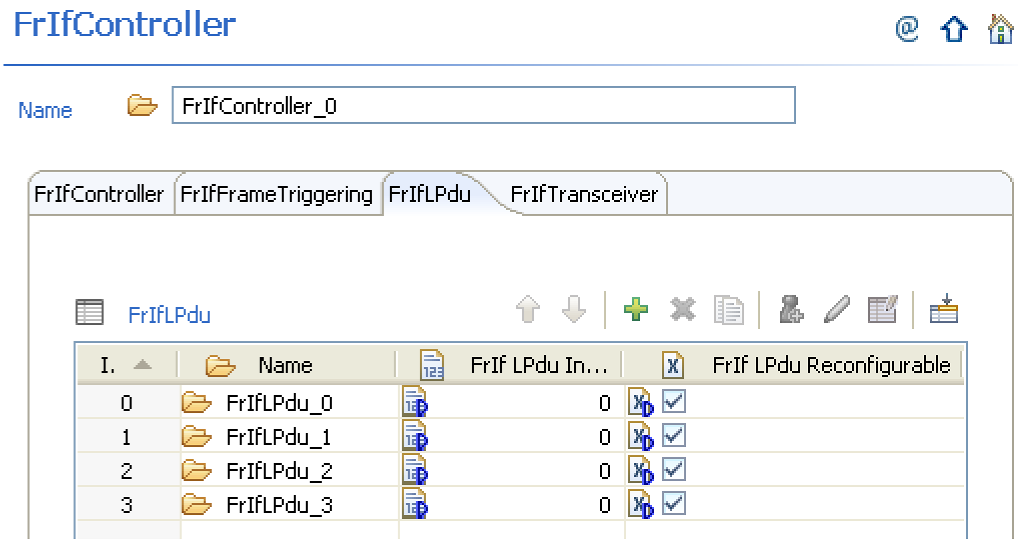
Description	Defines the index of the FrIf CC.
Class	Autosar Parameter
Range	0 to 31
Default	0
Source File	n/a
Source Representation	n/a

Table 4-16. FrIfCtrlRef

Description	References the appropriate FlexRay CC configured in the Fr container. Each FlexRay Interface multiple configuration set needs to be linked with one FlexRay multiple configuration set.
Class	Autosar Parameter
Range	n/a
Default	n/a
Source File	n/a
Source Representation	n/a

4.3.1.1 FrIfLPdu

FlexRay interface and FlexRay driver(s) communicate with each other using a LPdu index as a handle. Each (cycle, slot) pair that is used for transmission or reception on a cluster gets an own LPdu index assigned. That way, FlexRay Interface does not need to know about any hardware restrictions a certain controller might have. It is up to the FlexRay driver to map LPdu's to real hardware buffers. In fact the Tresos generator tool executes this mapping.

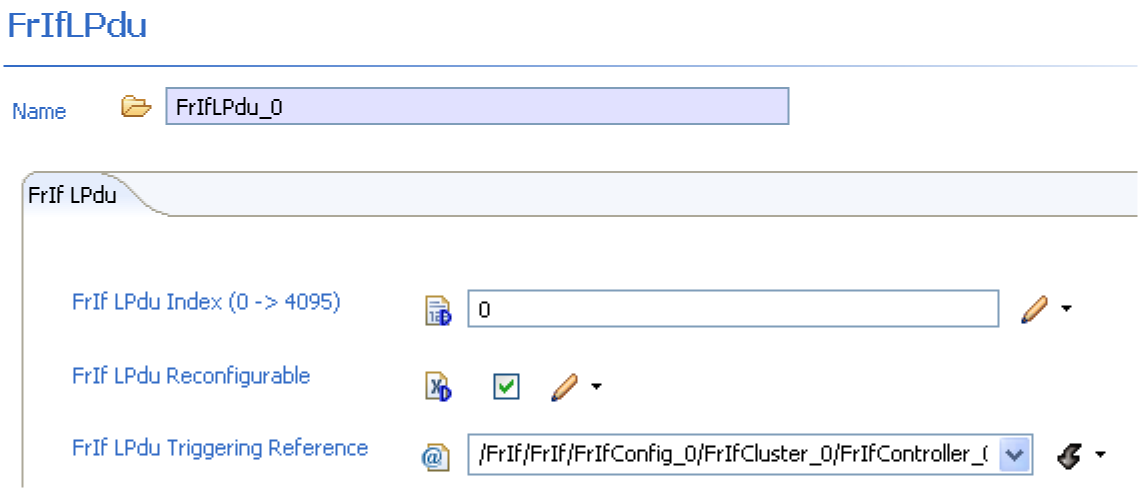


FrIfController

Name: FrIfController_0

I.	Name	FrIf LPdu In...	FrIf LPdu Reconfigurable
0	FrIfLPdu_0	0	☑
1	FrIfLPdu_1	0	☑
2	FrIfLPdu_2	0	☑
3	FrIfLPdu_3	0	☑

Figure 4-13. FrIfCluster Container - FrIfController - the List of the FrIfLPdu Subcontainers



FrIfLPdu

Name: FrIfLPdu_0

FrIf LPdu Index (0 -> 4095): 0

FrIf LPdu Reconfigurable: ☑

FrIf LPdu Triggering Reference: /FrIf/FrIf/FrIfConfig_0/FrIfCluster_0/FrIfController_(

Figure 4-14. FrIfCluster Container - FrIfController - FrIfLPdu Section

One or more LPdu's can be defined in the FrIfController container. The Tresos generator tool configures one FlexRay message buffer or one FIFO storage for each of these LPdu's.

Table 4-17. FrIfLPduIdx

Description	Identifies the LPdu in the interaction between FlexRay Interface and FlexRay Driver.
Class	Autosar Parameter
Range	0 to 4095
Default	0
Source File	Fr_PBcfg_[VariantName].c

Source Representation	`CONST(Fr_CCLPduInfoType, FR_APPL_CONST)`
	`FrIfMC0_Clst0_Ctrl0_LPduInfoCfgSet_PB[] =`
	`{`
	`  {FR_TRANSMIT_BUFFER, &FrIfMC0_Clst0_Ctrl0_LPdu0_Cfg_PB, 0U, TRUE, FALSE},`
	`  {FR_RECEIVE_BUFFER, &FrIfMC0_Clst0_Ctrl0_LPdu1_Cfg_PB, 1U, TRUE, FALSE},`
	`};`

Table 4-18. FrIfReconfigurable

Description	This parameter specifies that this LPdu is reconfigurable using FrIf_ReconfigLPdu. This means that this LPdu can be assigned to a different FrameTriggering at runtime. However, this reconfiguration is limited by hardware constraints. The direction of the LPdu cannot be reconfigured.
Class	Autosar Parameter
Range	True, False
Default	True
Source File	Fr_PBcfg_[VariantName].c

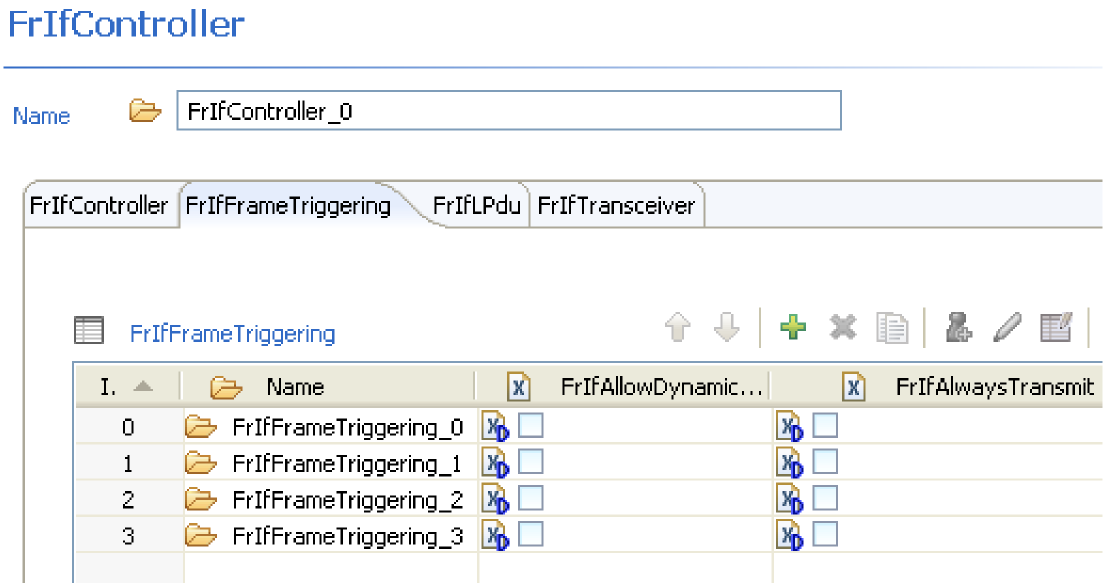
Source Representation	`CONST(Fr_CCLPduInfoType, FR_APPL_CONST)`
	`FrIfMC0_Clst0_Ctrl0_LPduInfoCfgSet_PB[] =`
	`{`
	`  {FR_TRANSMIT_BUFFER, &FrIfMC0_Clst0_Ctrl0_LPdu0_Cfg_PB, 0U, TRUE, TRUE},`
	`  {FR_RECEIVE_BUFFER, &FrIfMC0_Clst0_Ctrl0_LPdu1_Cfg_PB, 0U, FALSE,` **`FALSE`**`},`
	`};`

Table 4-19. FrIfVBTriggeringRef

Description	References to the assigned Frame triggering (to the relevant FrIfFrameTriggering container).
Class	Autosar Parameter
Range	n/a
Default	n/a
Source File	n/a
Source Representation	n/a

4.3.1.2 FrIfFrameTriggering

A frame triggering defines that a frame shall be sent in a certain (slot, base cycle, cycle repetition, channel) pattern.



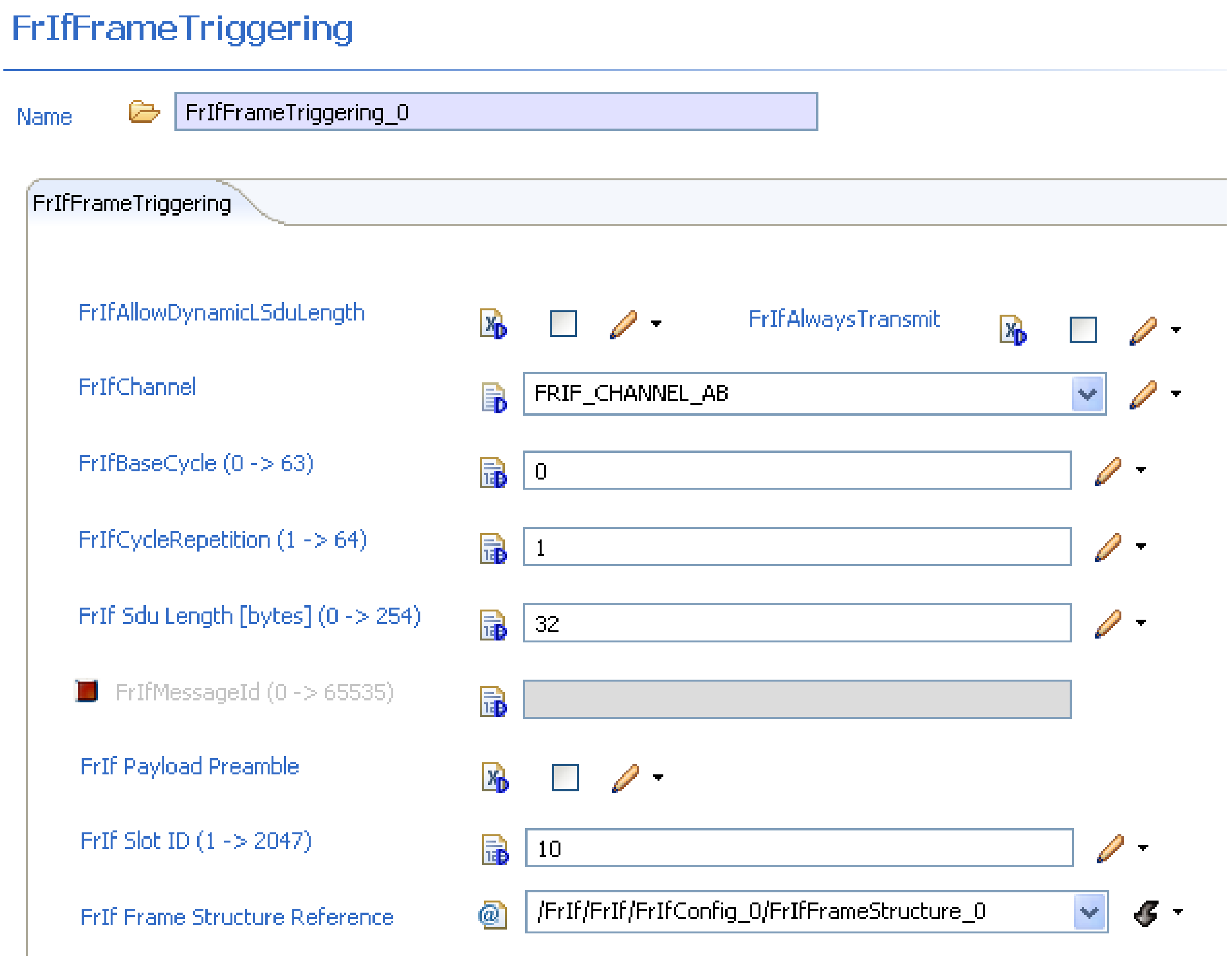
FrIfController

Name: FrIfController_0

FrIfFrameTriggering

I.	Name	FrIfAllowDynamic...	FrIfAlwaysTransmit
0	FrIfFrameTriggering_0	□	□
1	FrIfFrameTriggering_1	□	□
2	FrIfFrameTriggering_2	□	□
3	FrIfFrameTriggering_3	□	□

Figure 4-15. FrIfCluster Container - FrIfController - the List of the FrIfFrameTriggering Subcontainers



FrIfFrameTriggering

Name FrIfFrameTriggering_0

FrIfFrameTriggering

FrIfAllowDynamicLSduLength □ FrIfAlwaysTransmit □

FrIfChannel FRIF_CHANNEL_AB

FrIfBaseCycle (0 -> 63) 0

FrIfCycleRepetition (1 -> 64) 1

FrIf Sdu Length [bytes] (0 -> 254) 32

FrIfMessageId (0 -> 65535)

FrIf Payload Preamble □

FrIf Slot ID (1 -> 2047) 10

FrIf Frame Structure Reference /FrIf/FrIf/FrIfConfig_0/FrIfFrameStructure_0

Figure 4-16. FrIfCluster Container - FrIfController - FrIfFrameTriggering Section

Each FrIfLPdu container contains reference to one FrIfFrameTriggering container with additional configuration data. For successful message buffer configuration, the following parameters are required.

Table 4-20. FrIfAllowDynamicLSduLength

Description	Allows LPdu length reduction and indicates that the related CC buffer has to be reconfigured for the actual length and Header-CRC before transmission of the LPdu. It can be enabled only for virtual buffers configured for transmit operations.
Class	Autosar Parameter
Range	True, False
Default	False
Source File	Fr_PBcfg_[VariantName].c

Source Representation	`CONST(Fr_CCTransmitBufferConfigType, FR_APPL_CONST)`
	`FrIfMC0_Clst0_Ctrl0_LPdu1_Cfg_PB =`
	`{`
	`  62U, /* Transmit Frame ID */`
	`  0x03bfU, /* Frame Header CRC */`
	`  8U, /* Data Length in Words */`

Table 4-20. FrIfAllowDynamicLSduLength

	`TRUE,   /* Reception on channel A enabled */`
	`TRUE,   /* Reception on channel B enabled */`
	`FALSE,  /* Payload preamble disabled */`
	`TRUE,   /* Cycle counter filtering enabled */`
	`0U, /* Cycle counter filter match value */`
	`0x1fU, /* Cycle counter filter mask (repetition each 32 cycles) */`
	`TRUE` `  /* Dynamic payload length enabled */`
	`};`

The following parameter is not used by Tresos generator tool for the FlexRay driver

Table 4-21. FrIfAlwaysTransmit

Description	Defines whether the driver's API function `Fr_TransmitTxLPdu()` shall always be called for this LPdu.
Class	Autosar Parameter
Range	True, False
Default	False
Source File	n/a
Source Representation	n/a

Table 4-22. FrIfChannel

Description	Selects which FlexRay channel is used for given LPdu.
Class	Autosar Parameter
Range	FRIF_CHANNEL_AB, FRIF_CHANNEL_A, FRIF_CHANNEL_B
Default	FRIF_CHANNEL_AB
Source File	Fr_PBcfg_[VariantName].c

Source Representation	`CONST(Fr_CCTransmitBufferConfigType, FR_APPL_CONST)`
	`FrIfMC0_Clst0_Ctrl0_LPdu1_Cfg_PB =`
	`{`
	`62U, /* Transmit Frame ID */`
	`0x03bfU, /* Frame Header CRC */`
	`8U, /* Data Length in Words */`
	`TRUE``,   /* Reception on channel A enabled */`
	`TRUE``,   /* Reception on channel B enabled */`
	`FALSE,  /* Payload preamble disabled */`
	`TRUE,   /* Cycle counter filtering enabled */`
	`0U, /* Cycle counter filter match value */`
	`0x1fU, /* Cycle counter filter mask (repetition each 32 cycles) */`
	`TRUE   /* Dynamic payload length enabled */`
	`};`
	`CONST(Fr_CCReceiveBufferConfigType, FR_APPL_CONST)`
	`FrIfMC0_Clst0_Ctrl0_LPdu3_Cfg_PB =`
	`{`
	`63U, /* Receive Frame ID */`
	`4U, /* Data Length in Words */`
	`TRUE``,   /* Reception on channel A enabled */`
	`FALSE``,  /* Reception on channel B disabled */`
	`FALSE,  /* Cycle counter filtering disabled */`
	`0U, /* Cycle counter filter match value */`

Table 4-22. FrIfChannel

	`  0x00U /* Cycle counter filter mask (repetition each 1 cycles) */`
	`};`

Table 4-23. FrIfBaseCycle

Description	Contains the FlexRay Base Cycle used to transmit the FlexRay Frame
Class	Autosar Parameter
Range	0 to 63
Default	0
Source File	Fr_PBcfg_[VariantName].c

Source Representation	`CONST(Fr_CCTransmitBufferConfigType, FR_APPL_CONST)`
	`FrIfMC0_Clst0_Ctrl0_LPdu1_Cfg_PB =`
	`{`
	`  62U, /* Transmit Frame ID */`
	`  0x03bfU, /* Frame Header CRC */`
	`  8U, /* Data Length in Words */`
	`  TRUE,   /* Reception on channel A enabled */`
	`  TRUE,   /* Reception on channel B enabled */`
	`  FALSE,  /* Payload preamble disabled */`
	`  TRUE,   /* Cycle counter filtering enabled */`
	`  0U, /* Cycle counter filter match value */`
	`  0x1fU, /* Cycle counter filter mask (repetition each 32 cycles) */`
	`  TRUE   /* Dynamic payload length enabled */`
	`};`
	`CONST(Fr_CCReceiveBufferConfigType, FR_APPL_CONST)`
	`FrIfMC0_Clst0_Ctrl0_LPdu3_Cfg_PB =`
	`{`
	`  63U, /* Receive Frame ID */`
	`  4U, /* Data Length in Words */`
	`  TRUE,   /* Reception on channel A enabled */`
	`  FALSE,  /* Reception on channel B disabled */`
	`  FALSE,  /* Cycle counter filtering disabled */`
	`  0U, /* Cycle counter filter match value */`
	`  0x00U /* Cycle counter filter mask (repetition each 1 cycles) */`
	`};`

Table 4-24. FrIfCycleRepetition

Description	Contains the FlexRay Cycle Repetition used to transmit the FlexRay Frame.
Class	Autosar Parameter
Range	1,2,4,8,16,32,64
Default	1
Source File	Fr_PBcfg_[VariantName].c

Source Representation	`CONST(Fr_CCTransmitBufferConfigType, FR_APPL_CONST)`
	`FrIfMC0_Clst0_Ctrl0_LPdu1_Cfg_PB =`
	`{`
	`  62U, /* Transmit Frame ID */`
	`  0x03bfU, /* Frame Header CRC */`
	`  8U, /* Data Length in Words */`
	`  TRUE,   /* Reception on channel A enabled */`
	`  TRUE,   /* Reception on channel B enabled */`

Table 4-24. FrIfCycleRepetition

	FALSE, /* Payload preamble disabled */
	TRUE, /* Cycle counter filtering enabled */
	0U, /* Cycle counter filter match value */
	0x1fU, /* Cycle counter filter mask (repetition each 32 cycles) */
	TRUE /* Dynamic payload length enabled */
	};
	CONST(Fr_CCReceiveBufferConfigType, FR_APPL_CONST)
	FrIfMC0_Clst0_Ctrl0_LPdu3_Cfg_PB =
	{
	63U, /* Receive Frame ID */
	4U, /* Data Length in Words */
	TRUE, /* Reception on channel A enabled */
	FALSE, /* Reception on channel B disabled */
	FALSE, /* Cycle counter filtering disabled */
	0U, /* Cycle counter filter match value */
	0x00U /* Cycle counter filter mask (repetition each 1 cycles) */
	};

Table 4-25. FrIfLSduLength

Description	Specifies the payload length of the Frame. This parameter is required for validation if configured PDU's and update information fits into the Frame at configuration time [bytes].
Class	Autosar Parameter
Range	0 to 254
Default	32
Source File	Fr_PBcfg_[VariantName].c

Source Representation	CONST(Fr_CCTransmitBufferConfigType, FR_APPL_CONST)
	FrIfMC0_Clst0_Ctrl0_LPdu1_Cfg_PB =
	{
	62U, /* Transmit Frame ID */
	0x03bfU, /* Frame Header CRC */
	8U, /* Data Length in Words */
	TRUE, /* Reception on channel A enabled */
	TRUE, /* Reception on channel B enabled */
	FALSE, /* Payload preamble disabled */
	TRUE, /* Cycle counter filtering enabled */
	0U, /* Cycle counter filter match value */
	0x1fU, /* Cycle counter filter mask (repetition each 32 cycles) */
	TRUE /* Dynamic payload length enabled */
	};
	CONST(Fr_CCReceiveBufferConfigType, FR_APPL_CONST)
	FrIfMC0_Clst0_Ctrl0_LPdu3_Cfg_PB =
	{
	63U, /* Receive Frame ID */
	4U, /* Data Length in Words */
	TRUE, /* Reception on channel A enabled */
	FALSE, /* Reception on channel B disabled */
	FALSE, /* Cycle counter filtering disabled */
	0U, /* Cycle counter filter match value */
	0x00U /* Cycle counter filter mask (repetition each 1 cycles) */
	};

Table 4-26. FrIfPayloadPreamble

Description	Switching the Payload Preamble bit.

Table continues on the next page...

Table 4-26. FrIfPayloadPreamble (continued)

Class	Autosar Parameter
Range	True, False
Default	Flase
Source File	Fr_PBcfg_[VariantName].c

Source Representation	CONST(Fr_CCTransmitBufferConfigType, FR_APPL_CONST)
	FrIfMC0_Clst0_Ctrl0_LPdu1_Cfg_PB =
	{
	  62U, /* Transmit Frame ID */
	  0x03bfU, /* Frame Header CRC */
	  8U, /* Data Length in Words */
	  TRUE, /* Reception on channel A enabled */
	  TRUE, /* Reception on channel B enabled */
	  FALSE, /* Payload preamble disabled */
	  TRUE, /* Cycle counter filtering enabled */
	  0U, /* Cycle counter filter match value */
	  0x1fU, /* Cycle counter filter mask (repetition each 32 cycles) */
	  TRUE /* Dynamic payload length enabled */
	};

Table 4-27. FrIfSlotId

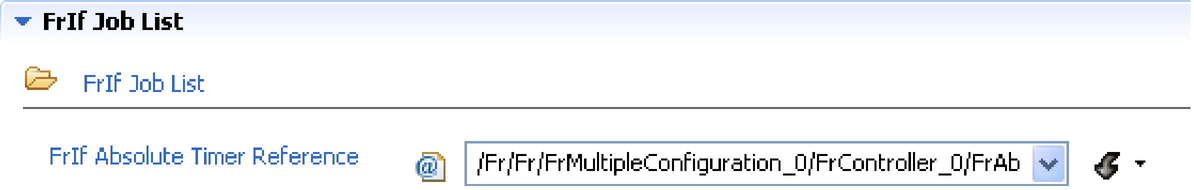
Description	Contains the FlexRay Slot ID used to transmit the FlexRay Frame.
Class	Autosar Parameter
Range	1 to 2047
Default	n/a
Source File	Fr_PBcfg_[VariantName].c

Source Representation	CONST(Fr_CCTransmitBufferConfigType, FR_APPL_CONST)
	FrIfMC0_Clst0_Ctrl0_LPdu1_Cfg_PB =
	{
	  62U, /* Transmit Frame ID */
	  0x03bfU, /* Frame Header CRC */
	  8U, /* Data Length in Words */
	  TRUE, /* Reception on channel A enabled */
	  TRUE, /* Reception on channel B enabled */
	  FALSE, /* Payload preamble disabled */
	  TRUE, /* Cycle counter filtering enabled */
	  0U, /* Cycle counter filter match value */
	  0x1fU, /* Cycle counter filter mask (repetition each 32 cycles) */
	  TRUE /* Dynamic payload length enabled */
	};
	CONST(Fr_CCReceiveBufferConfigType, FR_APPL_CONST)
	FrIfMC0_Clst0_Ctrl0_LPdu3_Cfg_PB =
	{
	  63U, /* Receive Frame ID */
	  4U, /* Data Length in Words */
	  TRUE, /* Reception on channel A enabled */
	  FALSE, /* Reception on channel B disabled */
	  FALSE, /* Cycle counter filtering disabled */
	  0U, /* Cycle counter filter match value */
	  0x00U /* Cycle counter filter mask (repetition each 1 cycles) */
	};

Table 4-28. FrIfFrameStructureRef

Description	References to the Construction Plan of the FlexRay Frame (to the relevant FrIfFrameStructure container).
Class	Autosar Parameter
Range	n/a
Default	n/a
Source File	n/a
Source Representation	n/a

4.3.2 FrIfCluster - FrIfController



FrIf Absolute Timer Reference /Fr/Fr/FrMultipleConfiguration_0/FrController_0/FrAb

Figure 4-17. FrIfCluster Container - FrIfJobList Section

A FlexRay job list is a list of FlexRay Communication jobs sorted according to their respective execution start time. FlexRay driver Tresos generator tool uses the job list configuration to analyze the reconfiguration possibilities. The `PREPARE_LPDU` action needs to be defined in FlexRay job list for each LPdu which shall be used for reconfiguration. Tresos generator tool analyses the start and end times of each single operation (transmission or reception) for each LPdu and it determines whether any physical resource(s) can be shared. Each single operation starts with `PREPARE_LPDU` action. The end times are calculated for

- transmission from
  - either time of TX_CONFIRMATION action
  - or time of physical data transmission
- reception from
  - either time of RECEIVE_AND_INDICATE action
  - or time of RECEIVE_AND_STORE action
  - or RX_CONFIRMATION action

Table 4-29. FrIfAbsTimerRef

Description	Reference to the absolute timer to be used to trigger the interrupt whose ISR contains the FlexRay Interface Job List function.
Class	Autosar Parameter
Range	n/a

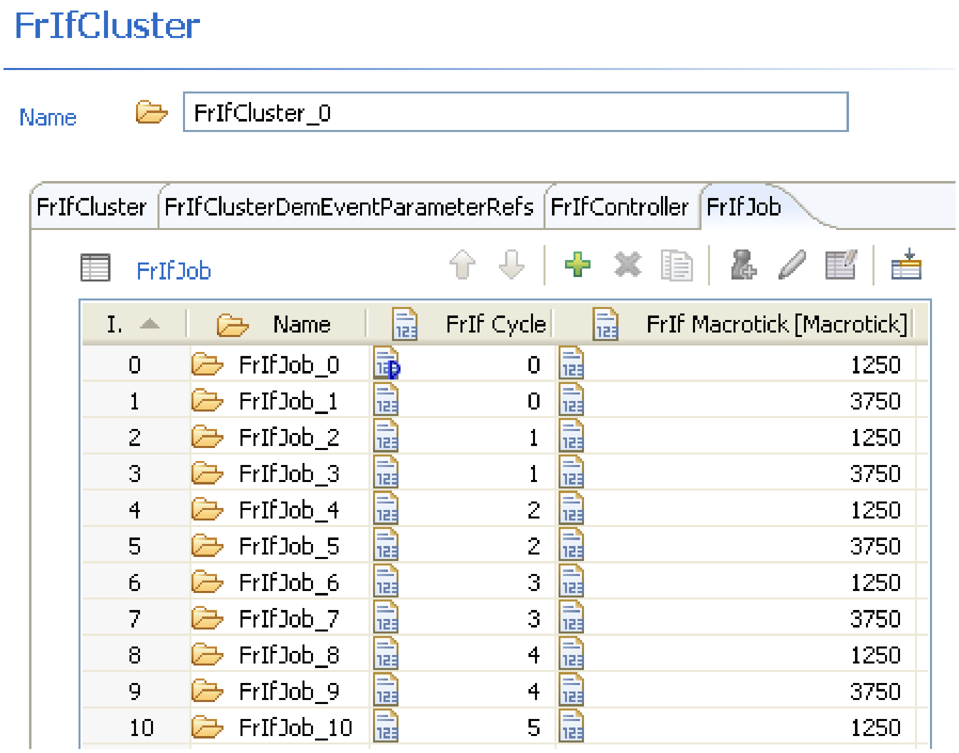
Table continues on the next page...

Table 4-29. FrIfAbsTimerRef (continued)

Default	n/a
Source File	n/a
Source Representation	n/a

4.3.2.1 FrIfJob

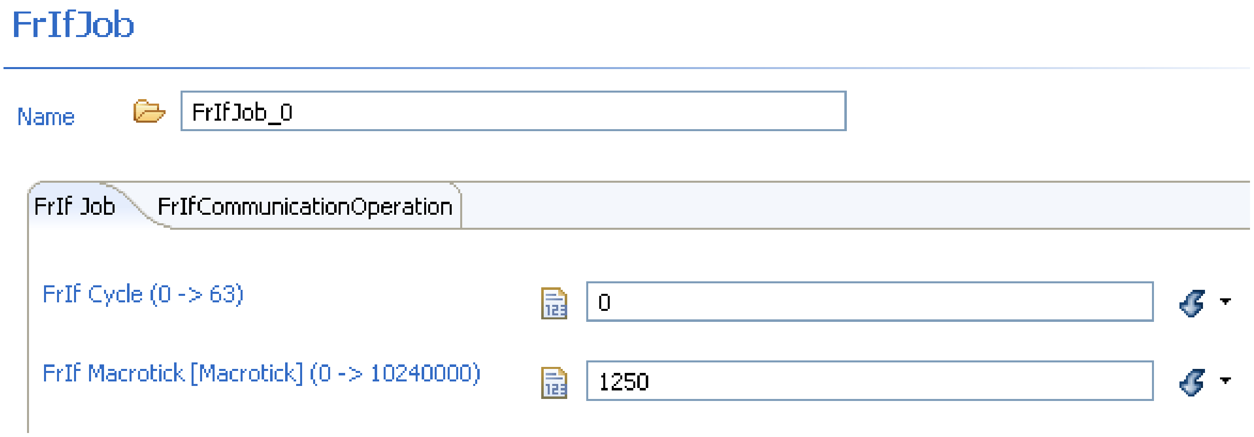
FlexRay job list is a list of FlexRay communication jobs sorted according to their respective execution start time. Each communication job contains the job start time and a list of communication operations sorted according to a communication operation index.



Name: FrIfCluster_0

I.	Name	FrIf Cycle	FrIf Macrotick [Macrotick]
0	FrIfJob_0	0	1250
1	FrIfJob_1	0	3750
2	FrIfJob_2	1	1250
3	FrIfJob_3	1	3750
4	FrIfJob_4	2	1250
5	FrIfJob_5	2	3750
6	FrIfJob_6	3	1250
7	FrIfJob_7	3	3750
8	FrIfJob_8	4	1250
9	FrIfJob_9	4	3750
10	FrIfJob_10	5	1250

Figure 4-18. FrIfCluster Container - FrIfJobList - the List of the FrIfJob Subcontainers



FrIfJob

Name FrIfJob_0

FrIf Cycle (0 -> 63) 0

FrIf Macrotick [Macrotick] (0 -> 10240000) 1250

Figure 4-19. FrIfCluster Container - FrIfJobList - FrIfJob Section

Table 4-30. FrIfCycle

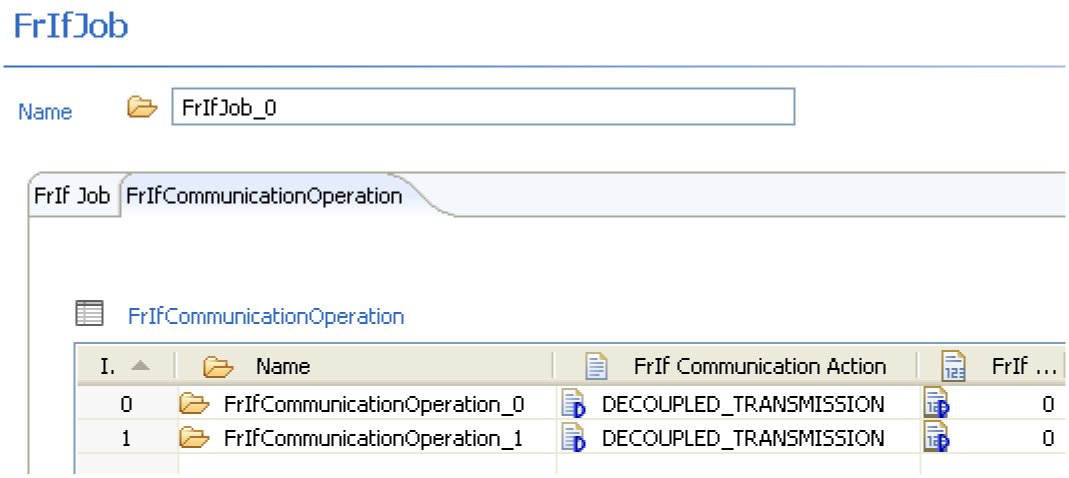
Description	Defines the FlexRay communication cycle in which this job is executed.
Class	Autosar Parameter
Range	0 to 63
Default	0
Source File	n/a
Source Representation	n/a

Table 4-31. FrIfMacrotick

Description	Defines macrotick offset in the communication cycle.
Class	Autosar Parameter
Range	0 to 10240000
Default	0
Source File	n/a
Source Representation	n/a

4.3.2.2 FrIfCommunicationOperation

The communication operations specify the actions to process within the communication job.



FrIfJob

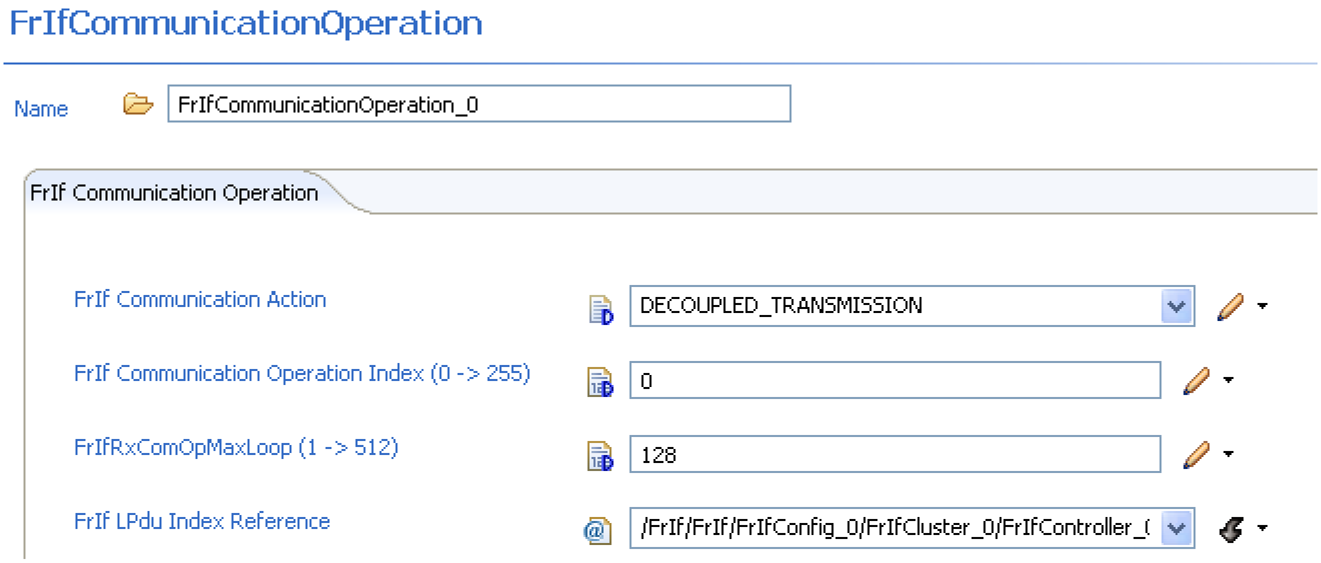
Name: FrIfJob_0

FrIf Job | FrIfCommunicationOperation

FrIfCommunicationOperation

I.	Name	FrIf Communication Action	FrIf ...
0	FrIfCommunicationOperation_0	DECOUPLED_TRANSMISSION	0
1	FrIfCommunicationOperation_1	DECOUPLED_TRANSMISSION	0

Figure 4-20. FrIfCluster Container - FrIfJobList - FrIfCommunicationOperation Subcontainers



FrIfCommunicationOperation

Name: FrIfCommunicationOperation_0

FrIf Communication Operation

FrIf Communication Action: DECOUPLED_TRANSMISSION

FrIf Communication Operation Index (0 -> 255): 0

FrIfRxComOpMaxLoop (1 -> 512): 128

FrIf LPdu Index Reference: /FrIf/FrIf/FrIfConfig_0/FrIfCluster_0/FrIfController_(

Figure 4-21. FrIfCluster Container - FrIfJobList - FrIfCommunicationOperation Section

Table 4-32. FrIfCommunicationAction

Description	Defines the action to be performed in the FlexRay Operation.
Class	Autosar Parameter
Range	DECOUPLED_TRANSMISSION, TX_CONFIRMATION, RECEIVE_AND_STORE, RX_INDICATION, RECEIVE_AND_INDICATE, PREPARE_LPDU
Default	DECOUPLED_TRANSMISSION
Source File	n/a
Source Representation	n/a

Table 4-33. FrIfCommunicationOperationIdx

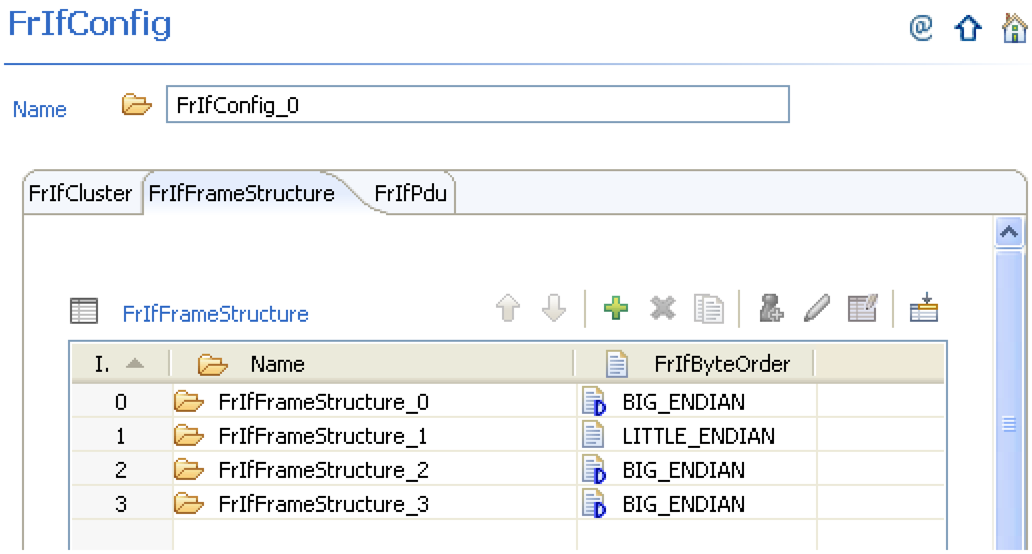
Description	Defines communication operation index, which determines the execution order of the communication operations within the communication job.
Class	Autosar Parameter
Range	0 to 255
Default	0
Source File	n/a
Source Representation	n/a

Table 4-34. FrIfLPduIdxRef

Description	Contains a reference to an LPdu that is associated with the communication action.
Class	Autosar Parameter
Range	n/a
Default	n/a
Source File	n/a
Source Representation	n/a

4.4 FrIfFrameStructure

The Frame structure specifies a Construction Plan how a frame is assembled with PDU's and their respective Update-Bits.



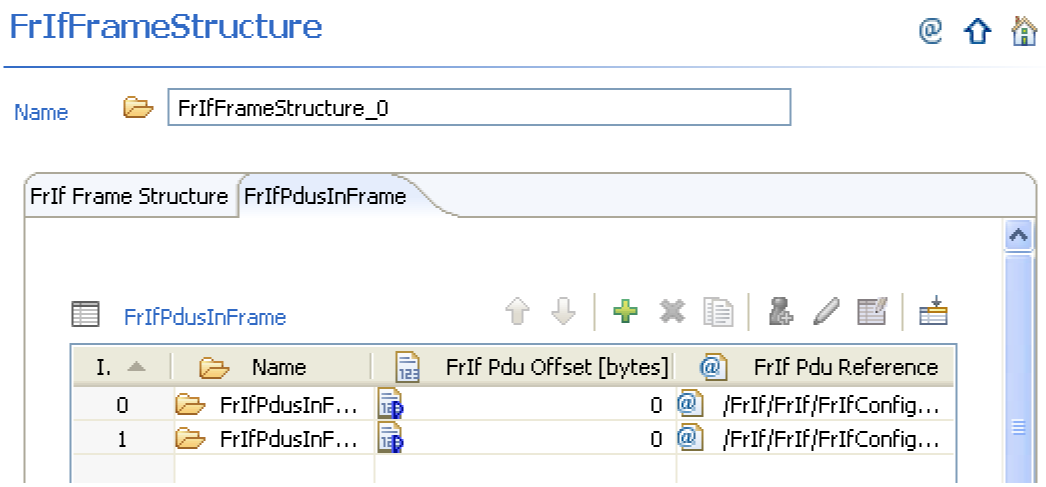
Name FrIfConfig_0

I.	Name	FrIfByteOrder
0	FrIfFrameStructure_0	BIG_ENDIAN
1	FrIfFrameStructure_1	LITTLE_ENDIAN
2	FrIfFrameStructure_2	BIG_ENDIAN
3	FrIfFrameStructure_3	BIG_ENDIAN

Figure 4-22. FrIfFrameStructure Container - The List of Subcontainers

4.4.1 FrIfPdusInFrame

This container holds all the information about a PDU in relevant FlexRay Frame.

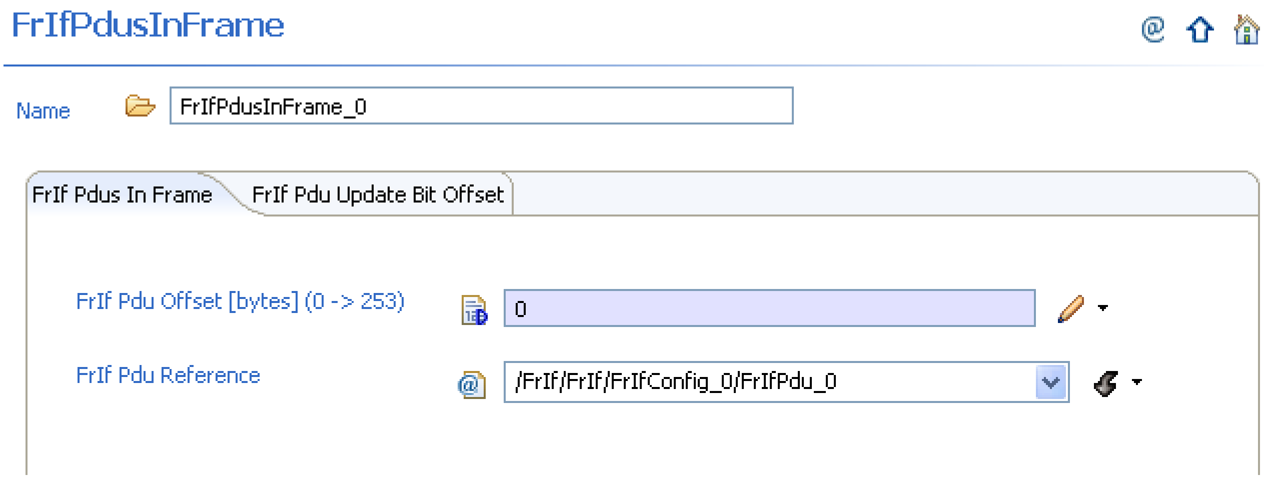


FrIfFrameStructure

Name: FrIfFrameStructure_0

I.	Name	FrIf Pdu Offset [bytes]	FrIf Pdu Reference
0	FrIfPdusInF...	0	/FrIf/FrIf/FrIfConfig...
1	FrIfPdusInF...	0	/FrIf/FrIf/FrIfConfig...

Figure 4-23. FrIfFrameStructure Container - The List of the FrIfPdusInFrame Subcontainers



FrIfPdusInFrame

Name: FrIfPdusInFrame_0

FrIf Pdu Offset [bytes] (0 -> 253): 0

FrIf Pdu Reference: /FrIf/FrIf/FrIfConfig_0/FrIfPdu_0

Figure 4-24. FrIfFrameStructure Container - FrIfPdusInFrame Subcontainer

Table 4-35. FrIfPduOffset

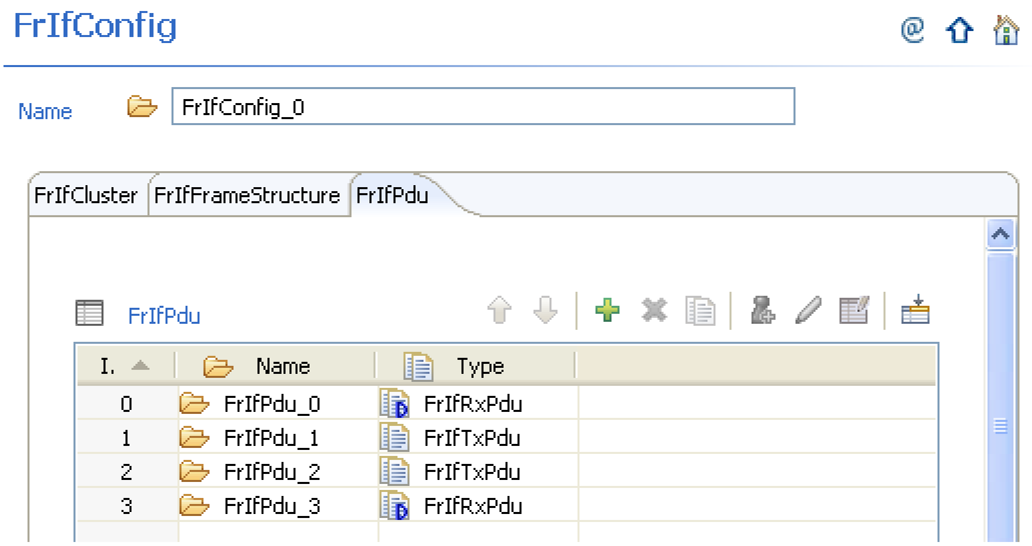
Description	The value specifies the offset of the PDU within the Frame [bytes].
Class	Autosar Parameter
Range	0 to 253
Default	0
Source File	n/a
Source Representation	n/a

Table 4-36. FrFrIfPduRef

Description	References to the local definition of a PDU (to the relevant FrIfPdu container).
Class	Autosar Parameter
Range	n/a
Default	n/a
Source File	n/a
Source Representation	n/a

4.5 FrIfPdu

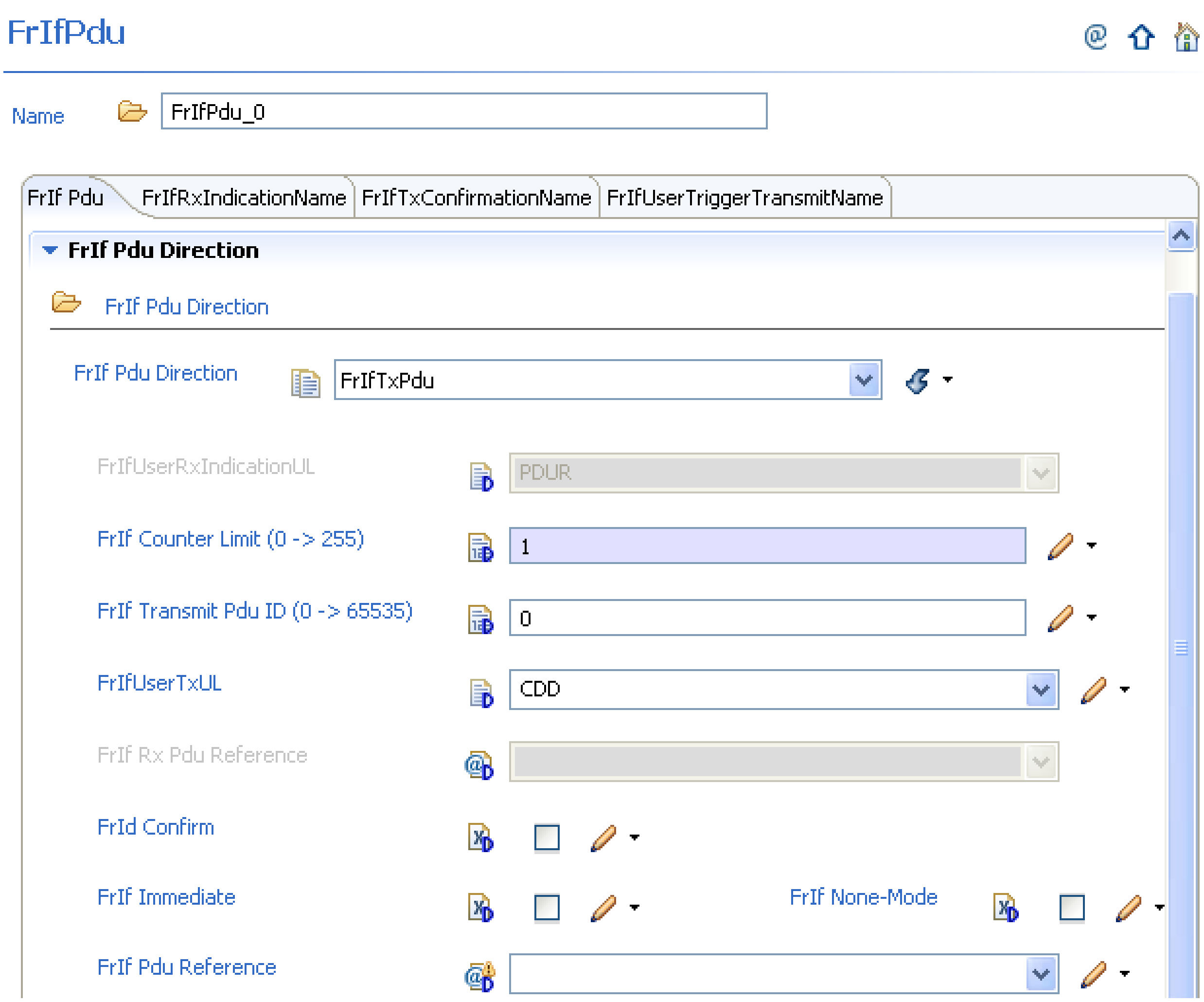
This container contains PDU's information. A PDU may be either a transmission PDU or a reception PDU. Only one parameter in FrIfPdu container is important for the FlexRay driver configuration.



Name: FrIfConfig_0

I.	Name	Type
0	FrIfPdu_0	FrIfRxPdu
1	FrIfPdu_1	FrIfTxPdu
2	FrIfPdu_2	FrIfTxPdu
3	FrIfPdu_3	FrIfRxPdu

Figure 4-25. FrIfPdu Container - The List of Subcontainers



FrIfPdu

Name: FrIfPdu_0

FrIf Pdu Direction

FrIf Pdu Direction

FrIf Pdu Direction: FrIfTxPdu

FrIfUserRxIndicationUL: PDUR

FrIf Counter Limit (0 -> 255): 1

FrIf Transmit Pdu ID (0 -> 65535): 0

FrIfUserTxUL: CDD

FrIf Rx Pdu Reference:

FrId Confirm: □

FrIf Immediate: □

FrIf None-Mode: □

FrIf Pdu Reference:

Figure 4-26. FrIfPdu Container - FrIfPduDirection

Table 4-37. FrIfPduDirection

Description	Specifies whether the PDU (and subsequently the whole LPdu) is transmitted or received.
Class	Autosar Parameter
Range	FrIfTxPdu, FrIfRxPdu
Default	FrIfRxPdu
Source File	Fr_PBcfg_[VariantName].c

Source Representation	CONST(Fr_CCLPduInfoType, FR_APPL_CONST)
	FrIfMC0_Clst0_Ctrl0_LPduInfoCfgSet_PB[] =
	{
	{FR_TRANSMIT_BUFFER, &FrIfMC0_Clst0_Ctrl0_LPdu0_Cfg_PB, 0U, TRUE, FALSE},
	{FR_RECEIVE_BUFFER, &FrIfMC0_Clst0_Ctrl0_LPdu1_Cfg_PB, 1U, TRUE, FALSE},
	{FR_RECEIVE_BUFFER, &FrIfMC0_Clst0_Ctrl0_LPdu2_Cfg_PB, 2U, TRUE, FALSE},
	{FR_TRANSMIT_BUFFER, &FrIfMC0_Clst0_Ctrl0_LPdu3_Cfg_PB, 3U, TRUE, FALSE},
	};

4.6 Physical Resources Reconfiguration

The Tresos generator tool tries to share one physical resource (i.e. message buffer) for several LPdu's (FlexRay frames) if the following conditions are met:

- Buffer reconfiguration functionality is enabled for FlexRay driver in FrGeneral/VendorSpecific container
- Given LPdu's are not assigned to any FIFO storage
- Given LPdu's are configured for operation in static segment of communication cycle
- PREPARE_LPDU communication operations are defined for these LPdu's in FrIf job list, see FrIfCluster - FrIfController for more information
- There are not enough physical resources available on FlexRay module for configured amount of LPdu's or Force Reconfiguration feature is enabled in FrController/VendorSpecific/ForceReconfiguration

Note

The message buffer sharing is limited for two FlexRay frames for one physical resource in the current version of Tresos generator tool.

4.7 Error and Warning Codes Description

In case of incorrect configuration the Tresos Generator Tool generates error or warning message.

4.7.1 Error Codes

Table 4-38. Error code 001 description

Error Code	001
Description	Referenced Fr CC ("FrIfFrCtrlRef" parameter) in the FrIf Controller configuration "[FrIfCC_configuration_path]" is not valid.
Solution	Correct value of given node to contain a valid reference to the Fr CC node.

Table 4-39. Error code 002 description

Error Code	002
Description	CC [x] Configuration "name" - Fr Controller is assigned to more than one FrIf cluster in one FrIf (multiple) configuration.
Solution	Assign the controller identified by number x to only one cluster.

Table 4-40. Error code 003 description

Error Code	003
Description	The FrIfController is not used in any FrIfCluster.
Solution	Check that the FrIfCluster container contains at least one FrIfController subcontainer.

Table 4-41. Error code 004 description

Error Code	004
Description	FrKeySlotId value is greater than the FrIfGNumberOfStaticSlots value.
Solution	Check whether the FrKeySlotId parameter is correctly configured to a slot in the static segment.

Table 4-42. Error code 005 description

Error Code	005
Description	Setting the Startup bit and not setting the Sync bit leads to an invalid configuration!
Solution	Check that FrPKeySlotUsedForSync parameter is not set to false in case of FrPKeySlotUsedForStartup is set to true.

Table 4-43. Error code 006 description

Error Code	006
Description	FrIfConfig/FrIfCluster/FrIfController/FrIfFrameTriggering/FrIfFrameStructureRef contains an invalid reference or refence is missing.
Solution	Check whether reference to the FrIfFrameStructure is valid.

Table 4-44. Error code 007 description

Error Code	007
Description	Selected Bit Rate value is not correctly configured.
Solution	Check and correct ChannelBitrate parameter in VendorSpecific containers according the Microcontroller Reference Manual (see MPC5748G Reference Manual [Table 2-3]).

Table 4-45. Error code 008 description

Error Code	008
Description	The list of FrAbsoluteTimer does not contain any Absolute timer, at least one is required.
Solution	Check number of configured absolute counters and their indices validity

Table 4-46. Error code 009 description

Error Code	009
Description	Too many configured absolute timers. The maximum number of absolute timers is 2.
Solution	Configure at most two timers.

Table 4-47. Error code 010 description

Error Code	010
Description	Configured absolute timer has an invalid FrAbsTimerIdx.
Solution	Configure FrAbsTimerIdx index of each relative timer to value 0 or 1.

Table 4-48. Error code 011 description

Error Code	011
Description	The index of each absolute timer should be lower than number of absolute timers in list of absolut timers (FrAbsoluteTimer).
Solution	Check whether the index is within range 0 to number of absolut timers.

Table 4-49. Error code 012 description

Error Code	012
Description	FrIfFrameTriggering reference in the “[FrIfLpdu_path]” node is not valid.
Solution	Correct value of given node to contain a valid reference in the “[FrIfLpdu_path]”.

Table 4-50. Error code 013 description

Error Code	013
Description	List of FIFOs (FrFrFifo in the FrController container) contains more than 2 FIFOs, only 2 of them are supported by the hardware.
Solution	Check that count of FIFOs is less or equal to 2.

Table 4-51. Error code 014 description

Error Code	014
Description	Both FIFOs are configured for the same channel
Solution	Check the configuration, only one FIFO is available for one channel.

Table 4-52. Error code 015 description

Error Code	015
Description	List of range (FrRange in the FrFifo container) does not contain any range filter.
Solution	Check that at least one range is defined.

Table 4-53. Error code 016 description

Error Code	016
Description	List of range (FrRange in the FrFifo container) contains more than 4 range filters.
Solution	Check that at most 4 ranges are defined.

Table 4-54. Error code 017 description

Error Code	017
Description	Parameter FrIfLSduLength is not equal to 2*FrIfGPayloadLengthStatic although the slot is configured in static segment.
Solution	Check the lengths of parameters FrIfLSduLength(in the FrIfFrameTriggering container) and FrIfGPayloadLengthStatic(in the FrIfCluster container).

Table 4-55. Error code 018 description

Error Code	018
Description	The FrIfAllowDynamicLSduLength parameter should not be configured for a frame in static segment of communication cycle.
Solution	Disable FrIfAllowDynamicLSduLength parameter for a frame configured for static segment of communication cycle.

Table 4-56. Error code 019 description

Error Code	019
Description	Parameter FrIfLSduLength is greater than 2*FrPPayloadLengthDynMax.
Solution	Check the lengths of parameters FrIfLSduLength(in the FrIfFrameTriggering container) and FrPPayloadLengthDynMax(in the FrIfCluster container).

Table 4-57. Error code 020 description

Error Code	020
Description	Some index of LPdus (FrIfLPduIdx in FrIfLPdu container) is not unique within all LPdus.
Solution	Check that all LPdu indices are unique.

Table 4-58. Error code 021 description

Error Code	021
Description	LPdu indices do not create continuous row beginning with 0.
Solution	Check that all LPdu indices constitute continuous row beginning with 0

Table 4-59. Error code 022 description

Error Code	022
Description	FrIfChannel is set to FRIF_CHANNEL_B(or FR_IF_CHANNEL_AB) but the controller is configured as single channel device. For a single channel device, the application can access and configure only the registers related to internal channel A.
Solution	If single channel mode is selected, configure FRIF_CHANNEL_A.

Table 4-60. Error code 023 description

Error Code	023
Description	The SingleChannelModeEnabled is enabled but FrPWakeupChannel is set to FR_CHANNEL_B. For a single channel device, the application can access and configure only the registers related to internal channel A.
Solution	If SingleChannelModeEnabled parameter is enabled, configure FrPWakeupChannel to FR_CHANNEL_A.

Table 4-61. Error code 024 description

Error Code	024
Description	The FrPChannels is set to FR_CHANNEL_A(FR_CHANNEL_B) only but FrPWakeupChannel is set to FR_CHANNEL_B(FR_CHANNEL_A).
Solution	Modify the FrPWakeupChannel parameter to have the same value as the FrPChannels parameter.

Table 4-62. Error code 025 description

Error Code	025
Description	FrIfChannel is set to FRIF_CHANNEL_B but the controller is configured as connected to FR_CHANNEL_A only.
Solution	Configure the FrIfChannel parameter to FRIF_CHANNEL_A or check the FrPChannels parameter.

Table 4-63. Error code 026 description

Error Code	026
Description	FrIfChannel is set to FRIF_CHANNEL_A but the controller is configured as connected to FR_CHANNEL_B only.
Solution	Configure the FrIfChannel parameter to FRIF_CHANNEL_B or check the FrPChannels parameter.

Table 4-64. Error code 027 description

Error Code	027
Description	The SingleChannelModeEnabled parameter is enabled but FrPChannels is set to FR_CHANNEL_AB.
Solution	Check whether the controller channels are correctly configured (parameters SingleChannelModeEnabled and FrPChannels).

Table 4-65. Error code 028 description

Error Code	028
Description	FrRangeMax is lower than FrRangeMin.
Solution	Verify range configuration.

Table 4-66. Error code 029 description

Error Code	029
Description	LPdu is configured for dynamic segment but FrIfReconfigurable is set.
Solution	Check that reconfiguration is disabled for dynamic segment.

Table 4-67. Error code 030 description

Error Code	030
Description	FrRange does not contain any acceptance range.
Solution	Check that at least one acceptance range is defined.

Table 4-68. Error code 031 description

Error Code	031
Description	LPdu will be received into the FIFO A but the FrIfReconfigurable is set.
Solution	Check which LPdus are received into the FIFO A and check that they are not reconfigurable.

Table 4-69. Error code 032 description

Error Code	032
Description	LPdu will be received into the FIFO B but the FrIfReconfigurable is set.
Solution	Check which LPdus are received into the FIFO B and check that they are not reconfigurable.

Table 4-70. Error code 033 description

Error Code	033
Description	FrIfCommunicationOperationIdx is not unique.
Solution	Check that all Communication Operation indices (FrIfCommunicationOperationIdx) are unique, separately for each Job.

Table 4-71. Error code 034 description

Error Code	034
Description	Not enough resources for all LPdus even after reconfiguration.
Solution	Consider the possibility of reconfiguration for another LPdus or reduce number of LPdus.

Table 4-72. Error code 035 description

Error Code	035
Description	Not enough resources for all LPdus.
Solution	Try enable the reconfiguration (ForceReconfiguration for global reconfiguration enable and FrIfReconfigurable for each different LPdu)

Table 4-73. Error code 036 description

Error Code	036
Description	Parameter FrIfCycleRepetition contains unsupported value.
Solution	Verify the FrIfCycleRepetition parameter (possible values: 1,2,4,8,16,32,64).

Table 4-74. Error code 037 description

Error Code	037
Description	FrControllerDemEventParameterRefs/FrDemCtrlTestResultRef has no reference to DemEventID
Solution	Select reference to DEM plugin for FrControllerDemEventParameterRefs/FrDemCtrlTestResultRef parameter.

Table 4-75. Error code 038 description

Error Code	038
Description	FrIfChannel is set to FRIF_CHANNEL_AB but the controller is configured as connected to FR_CHANNEL_B only. Transmission or reception in the dynamic segment will not work.
Solution	In the dynamic segment CC use only channel A in case of both channels are configured. Configure FrIfChannels to FRIF_CHANNEL_B only.

Table 4-76. Error code 039 description

Error Code	039
Description	SingleChannelModeEnabled is enabled but the FrFifo container contains more than 1 FIFO, only 1 FIFO is supported when the controller is configured as single channel device.
Solution	Reduce number of FIFOs in the FrFifo container to one FIFO only.

Table 4-77. Error code 040 description

Error Code	040
Description	SingleChannelModeEnabled is enabled but FrChannels is set to FR_CHANNEL_B. For a single channel device, the application can access and configure only the registers related to internal channel A.
Solution	Configure the FrChannels parameter in the FrFifo container to FR_CHANNEL_A.

Table 4-78. Error code 041 description

Error Code	041
Description	The FrFifo container contains more than 1 FIFO but FrPChannels is set to FR_CHANNEL_A(FR_CHANNEL_B) only. Only one FIFO per channel is supported by the hardware.
Solution	Reduce number of FIFOs in the FrFifo container to one FIFO only.

Table 4-79. Error code 042 description

Error Code	042
Description	Wrong channel is assigned to FIFO. FrChannels is configured to FR_CHANNEL_A(FR_CHANNEL_B) but FrPChannels is configured to FR_CHANNEL_B(FR_CHANNEL_A).
Solution	Configure the FrChannels parameter in the FrFifo container to the same value as is configured in FrPChannels.

Table 4-80. Error code 043 description

Error Code	043
Description	Maximum FIFO depth 255 entries was exceeded!.
Solution	Reduce FrFifoDepth either for FIFO A or FIFO B. **Note:** FIFO depth can be configured up to 255 entries for each FIFO. In case of both FIFO A and FIFO B are configured total FIFO depth can not exceed 255 entries.

Table 4-81. Error code 044 description

Error Code	044
Description	Some index of CC (FrCtrlIdx in the FrController container) is not unique.
Solution	Check that all CC indices are unique.

Table 4-82. Error code 045 description

Error Code	045
Description	CC indices (FrCtrlIdx in the FrController container) do not create continuous row beginning with 0.
Solution	Check that all CC indices constitute continuous row beginning with 0.

Table 4-83. Error code 046 description

Error Code	046
Description	FrIfClstIdx indices do not create zero-based consecutive sequence within the listed container.
Solution	Check that all FrIfClstIdx indices constitute continuous sequence beginning with 0.

Table 4-84. Error code 047 description

Error Code	047
Description	FrIfCtrlIdx indices do not create zero-based consecutive sequence within the listed container.
Solution	Check that all FrIfCtrlIdx indices constitute continuous sequence beginning with 0.

Table 4-85. Error code 5xx description

Error Code	5xx

Table continues on the next page...

Table 4-85. Error code 5xx description (continued)

Description	Internal error
Solution	If this error is shown, some unexpected situation occurred. Please contact support with following information: • Tresos Studio version • MCAL release version (best to send concrete Fr and FrIf plugins) • Error number and description • FrIf.xdm and Fr.xdm (located in the project\config folder)

Table 4-86. Error code 800 description

Error Code	800
Description	The length of gdNit must be equal to [gMacroPerCycle - gdSymbolWindow - (gNumberOfStaticSlots * gdStaticSlot) - (gNumberOfMinislots * gdMinislot)].
Solution	Verify configuration prameters: gdNit, gMacroPerCycle, gdSymbolWindow, gNumberOfStaticSlots, gdStaticSlot, gNumberOfMinislots, gdMinislot.

4.7.2 Warning Codes

Table 4-87. Warning code 109 description

Warning code	109
Description	Operation PREPARE_LPDU occurs 2 times in a row without physical use of the virtual buffer.
Solution	Check the configuration of FrIfFrameTriggering container for the corresponding virtual buffer.

Table 4-88. Warning code 110 description

Warning code	110
Description	Physical use of virtual buffer occurs two times without PREPARE_LPDU operation between them.
Solution	Add PREPARE_LPDU operation before each use of the virtual buffer.

Table 4-89. Warning code 111 description

Warning code	111
Description	Operation DECOUPLED_TRANSMISION is configured after physical use of virtual buffer.
Solution	DECOUPLED_TRANSMISSION operation should be configured before the physical data transmission from the virtual buffer.

Table 4-90. Warning code 112 description

Warning code	112
Description	Operation TX_CONFIRMATION is configured before physical use of virtual buffer.
Solution	Configure TX_CONFIRMATION operation after the physical data transmission from virtual buffer.

Table 4-91. Warning code 113 description

Warning code	113
Description	Some receive operation (RECEIVE_AND_INDICATE, RECEIVE_AND_STORE or RX_INDICATION) is configured before physical data reception into relevant virtual buffer.
Solution	Configure RECEIVE_AND_INDICATE, RECEIVE_AND_STORE or RX_INDICATION operation after the physical data reception into the virtual buffer.

Table 4-92. Warning code 114 description

Warning code	114
Description	FrIfFrameTriggeringDemEventParameterRefs/FrIfDemFTSlotStatusRef of the 'FrIfFrameTriggering_x' has no reference to DemEventID.
Solution	Configure reference to DEM plugin for FrIfFrameTriggeringDemEventParameterRefs/FrIfDemFTSlotStatusRef parameter or disable FrIfFrameTriggeringDemEventParameterRefs/FrIfDemFTSlotStatusRef configuration parameter for affected FrIfFrameTriggering container.

Table 4-93. Warning code 115 description

Warning code	115
Description	Dem reporting for the 'FrIfFrameTriggering_x' is disabled by the /FrGeneral/VendorSpecific/FrDisableDemReportErrorStatus configuration parameter.
Solution	Disable FrIfFrameTriggeringDemEventParameterRefs/FrIfDemFTSlotStatusRef configuration parameter for affected FrIfFrameTriggering container or enable DEM errors/events reporting by the /FrGeneral/VendorSpecific/FrDisableDemReportErrorStatus configuration parameter.

Table 4-94. Warning code 301 description

Warning code	301
Description	The SystemMemoryAccessTimeOut parameter is not correctly configured.
Solution	Change value to fulfill the following equation 0≤ SystemMemoryAccessTimeOut ≤0.45*fchi-8 where fchi is SystemFrequency parameter in the FrGeneral/VendorSpecific container.

Table 4-95. Warning code 302 description

Warning code	302
Description	There is no LPdu assigned for the Key Slot transmission.
Solution	Verify that one of LPdus is assigned for the Key Slot.

Table 4-96. Warning code 303 description

Warning code	303
Description	FrIfChannel is set to FRIF_CHANNEL_AB but transmission or reception in the dynamic segment on FlexRay channel B will be ignored.
Solution	In the dynamic segment configure FrIfChannel parameter to FRIF_CHANNEL_A or FRIF_CHANNEL_B only.

Table 4-97. Warning code 304 description

Warning code	304
Description	FrIfChannel is set to FRIF_CHANNEL_AB but the controller is configured as connected to FR_CHANNEL_A only. Transmission or reception on FlexRay channel B will be ignored.
Solution	Configure FrIfChannel parameter to FRIF_CHANNEL_A or change the FrPChannels parameter to FR_CHANNEL_AB.

Table 4-98. Warning code 305 description

Warning code	305
Description	FrIfChannel is set to FRIF_CHANNEL_AB but the controller is configured as connected to FR_CHANNEL_B only. Transmission or reception on FlexRay channel A will be ignored.
Solution	Configure FrIfChannel parameter to FRIF_CHANNEL_B or change the FrPChannels parameter to FR_CHANNEL_AB.

Table 4-99. Warning code 306 description

Warning code	306
Description	Lpdu with FrIfLPduIdx = xx was assigned to FIFO A but there is another LPdu with FrIfLPduIdx = yy already assigned to FIFO A in the FrIfCluster/FrIfController/FrIfLPdu.
Solution	Assigning more LPdus passing FIFO A criteria is redundant and has no functional effect. Configuration of one LPdu passing FIFO A criteria is sufficient.

Table 4-100. Warning code 307 description

Warning code	307
Description	Lpdu with FrIfLPduIdx = xx was assigned to FIFO B but there is another LPdu with FrIfLPduIdx = yy already assigned to FIFO B in the FrIfCluster/FrIfController/FrIfLPdu.
Solution	Assigning more LPdus passing FIFO B criteria is redundant and has no functional effect. Configuration of one LPdu passing FIFO B criteria is sufficient.

Table 4-101. Warning code 308 description

Warning code	308
Description	There is no LPdu matching FIFO A criteria for FrIfCluster/FrIfController/FrIfLPdu.
Solution	Configure one LPdu passing FIFO A criteria to generate FIFO A configuration.

Table 4-102. Warning code 309 description

Warning code	309
Description	There is no LPdu matching FIFO B criteria for FrIfCluster/FrIfController/FrIfLPdu.
Solution	Configure one LPdu passing FIFO B criteria to generate FIFO B configuration.

Table 4-103. Warning code 310 description

Warning code	310
Description	Too much controllers were configured for FrMultipleConfiguration/Fr_Config_() multiple configuration. Hardware device supports maximum "n" FlexRay controller(s).
Solution	Reduce number of configured communication controllers for affected multiple configuration not to be higher than number of communication controllers available on the hardware.

Table 4-104. Warning code 311 description

Warning code	311
Description	Too much controllers were configured for FrMultipleConfiguration/Fr_Config_() multiple configuration. Driver is configured to support "FrNumCtrlSupported" FlexRay controller(s).
Solution	Increase number of Fr driver supported controllers by changing the FrNumCtrlSupported configuration parameter or reduce number of configured communication controllers for affected multiple configuration not to be higher than the FrNumCtrlSupported configuration parameter.

Table 4-105. Warning code 312 description

Warning code	312
Description	Too much controllers are required to be supported by the driver. Driver will support maximum "n" FlexRay controller(s) available on the hardware.
Solution	Reduce number of supported controllers by changing the FrNumCtrlSupported configuration parameter not to be higher than number of communication controllers available on the hardware.

Table 4-106. Warning code 313 description

Warning code	313
Description	CCBaseAddress is not unique within one multiple configuration for the following controller.
Solution	Change CCBaseAddress to be unique for each controller within one multiple configuration to avoid problem that both configurations will configure the same FlexRay peripheral.

Table 4-107. Warning code 316 description

Warning code	316
Description	Lpdu with FrIfLPduIdx is a transmit Lpdu and its frame ID is in FIFOA ID range.
Solution	Lpdu with FrIfLPduIdx is a transmit Lpdu and its freame ID should not be in FIFOA ID range.

Table 4-108. Warning code 317 description

Warning code	317
Description	Lpdu with FrIfLPduIdx is a transmit Lpdu and its frame ID is in FIFOB ID range.
Solution	Lpdu with FrIfLPduIdx is a transmit Lpdu and its freame ID should not be in FIFOB ID range.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.