Document Title	Specification of Cellular Vehicle-2-X Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	1030

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R23-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2023-11-23	R23-11	AUTOSAR Release Management	• No content changes
2022-11-24	R22-11	AUTOSAR Release Management	• Initial release

Disclaimer

Contents

1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the AUTOSAR Basic Software module Cellular V2X Driver.

In the AUTOSAR Layered Software Architecture, the Cellular V2X driver belongs to the Microcontroller Abstraction Layer if the Cellular V2X controller is on-Chip type (internal), while the Cellular V2X driver belongs to Hardware Abstraction layer if the cellular V2X controller is off-chip type (external).

This indicates the main task of the Cellular V2X driver,which is:
Provide to the upper layer (Ethernet Interface for example) a hardware independent interface comprising multiple equal controllers. This interface shall be uniform for all controllers. Thus, the upper layer (Ethernet Interface for example) may access the underlying bus system in a uniform manner. The interface provides functionality for initialization, configuration and data transmission and facilities to manage/observe the lifecycle of the hardware. The configuration of the Cellular V2X Driver however is bus specific, since it takes into account the specific features of the wireless communication controller.

A single Cellular V2X driver module supports only one type of Cellular V2X hardware. The Cellular V2X driver's prefix requires a unique namespace. The Ethernet Interface can access different controller types using different Cellular V2X drivers using this prefix. The decision which driver to use to access a particular controller is a configuration parameter of the Ethernet Interface.

Figure 1 depicts an example of the lower part of the Cellular V2X stack. One Ethernet Interface can access several Cellular V2X hardware units, using several Cellular V2X drivers.

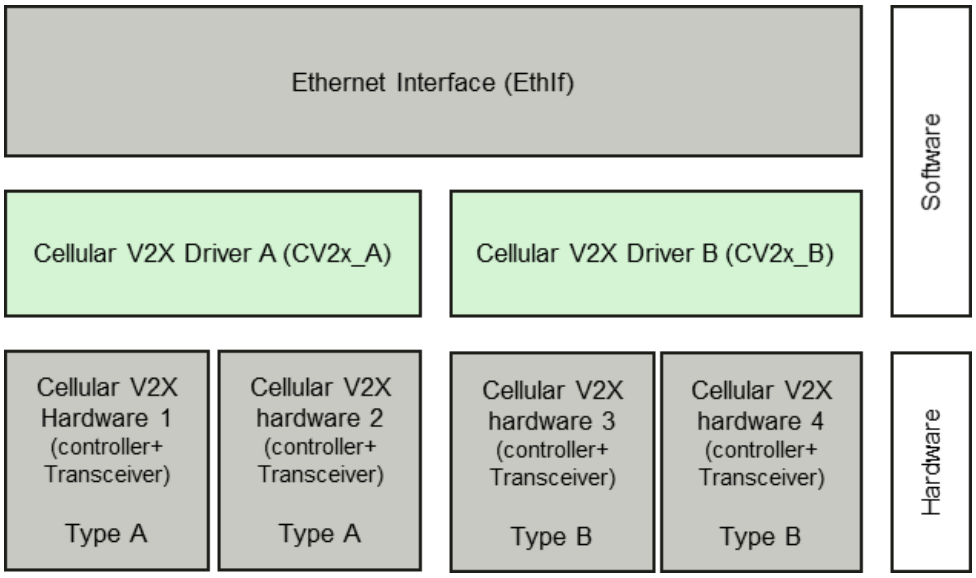

Figure 1.1: Example of the lower part of Cellular V2X Stack

Note:

1. Typically, Cellular V2X hardware includes both Cellular V2X RF transceiver and Cellular V2X Controller. There is no separated Cellular V2X transceiver/controller in the market, and cellular V2X transceiver is not controlled directly by ECU, therefore transceiver driver for Cellular V2X is not needed. In order to keep the naming consistent in AUTOSAR, "controller" is also used in this document to present the Cellular V2X hardware.
2. The Cellular V2X driver is specified in a way that allows for object code delivery of the code module, following the "one-fits-all" principle, i.e. the entire configuration of the Ethernet Interface can be carried out without modifying any source code. Thus, the configuration of the Cellular V2X driver can be carried out largely without detailed knowledge of the Cellular V2X driver software.

The bases for this document are the Chinese LTE-V2X based standards [1] [2]. It is assumed that the reader is familiar with these standards.

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the Cellular V2X driver module that are not included in the [3, AUTOSAR glossary].

Abbreviation / Acronym:	Description:
CBR	Channel Busy Ratio
CCSA	China Communications Standards Association
CV2x	Cellular Vehicle-2-X Driver
DSMP	Dedicated Short Message Protocol
IP	Internet protocol
LTE-V2X	Long Term Evolution based Vehicle to Everything
V2X	Vehicle to Everything
NTCAS	National Technical Committee of Auto Standardization
PC5	The reference point between the UEs (User equipment) used for control and user plane for ProSe (Proximity-based Services) Direct Communication for V2X Service
PPPP	ProSe Per-Packet Priority

3 Related documentation

3.1 Input documents & related standards and norms

[1] YD/T 3707-2020:Technical requirements of network layer of LTE-based vehicular communication
http://www.ccsa.org.cn/

[2] YD/T 3756-2020:Technical requirement of vehicle terminal for LTE-based vehicular communication
http://www.ccsa.org.cn/

[3] Glossary
AUTOSAR_FO_TR_Glossary

[4] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[5] Specification of Ethernet Driver
AUTOSAR_CP_SWS_EthernetDriver

[6] Specification of Ethernet Interface
AUTOSAR_CP_SWS_EthernetInterface

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [4], which is also valid for Cellular V2X Driver.

Thus, the specification SWS BSW General [4] shall be considered as additional and required specification for Cellular V2X Driver.

4 Constraints and assumptions

4.1 Limitations

- Cellular V2X Driver supports LTE-V2X PC5 only as defined by NTCAS and CCSA [1] [2]. Other cellular based wireless communication (e.g. LTE Uu interface) can be extended in future release of AUTOSAR standard.
- CV2x module support non-IP (i.e. DSMP) transmission only and mainly focus on broadcast based packet transport services in R22-11.
- It is not possible to transmit data, which exceeds the available buffer size of the used controller.
- Common parameters for access layer in cellular V2X hardware is usually preconfigured, thus common parameter setting(i.e. Transmit power, Center Frequency) is not supported in this release.
- The Microcontroller Abstraction Layer Multi-Core Distribution Concept is implemented as "draft" in this software specification. Refer to chapter 10 for more information.

4.2 Applicability to car domains

This specification is applicable to all car domains.

5 Dependencies to other modules

This chapter lists the modules interacting with the Cellular V2X Driver module.

Modules that use Cellular V2X Driver module:

- Ethernet Interface (EthIf)

5.1 Driver Services

[CP_SWS_CV2x_00001]{DRAFT} ⌈If the Cellular V2X controller is on-chip, the Cellular V2X Driver module shall not use any service of other drivers.⌋*()*

[CP_SWS_CV2x_00002]{DRAFT} ⌈If an off-chip Cellular V2X controller is used, the Wireless Ethernet driver shall use services of other MCAL drivers (e.g. SPI, DIO).⌋*()*

Note: In this case, the Cellular V2X Driver is not any more part of the Microcontroller Abstraction Layer but put part of the ECU abstraction layer. Therefore, it is theoretically allowed to use any Microcontroller Abstraction layer driver it needs.

Implementation hint: If the Cellular V2X driver uses services of other MCAL drivers (e.g. SPI, DIO), it must be ensured that these drivers are up and running before initializing the Wireless Ethernet driver.

[CP_SWS_CV2x_00003]{DRAFT} ⌈All the Cellular V2X Driver interfaces shall be implemented in a non-blocking manner. In cases where the action can be performed immediately and automically, the confirmation is reported in the request function's return code. Alternatively, the initiation of an action is performed by a call to a "request" function and the result of the action is reported by a corresponding "confirm" callback.⌋*()*

6 Requirements Tracing

Requirement	Description	Satisfied by
[CP_SRS_CnV2X_-00301]	The Access layer of Chinese V2X Communication shall be compliant to CCSA specification of Air Interface for LTE-based Vehicular Communication	[CP_SWS_CV2x_00015] [CP_SWS_CV2x_00017] [CP_SWS_CV2x_00019] [CP_SWS_CV2x_00021] [CP_SWS_CV2x_00023] [CP_SWS_CV2x_00024] [CP_SWS_CV2x_00025] [CP_SWS_CV2x_00026] [CP_SWS_CV2x_00027] [CP_SWS_CV2x_00031]
[SRS_BSW_00487]	Errors for module initialization shall follow a naming rule	[CP_SWS_CV2x_01064] [CP_SWS_CV2x_01070]

Table 6.1: RequirementsTracing

7 Functional specification

The Cellular V2X driver provides communications by accessing the Cellular V2X radio and enables Chinese V2X service. On transmission, the driver writes the packet into an appropriate buffer inside the Cellular V2X driver, on packet reception the Cellular V2X driver calls the receive packet callback function with the packet content passed in the argument.

7.1 Cellular V2X BSW stack

As part of the AUTOSAR Layered Software Architecture, the Cellular V2X BSW modules also form a layered software stack. To implement V2X services, the Ethernet Interface (EthIf) module can access one or several controllers using the Cellular V2X Driver layer, which can be made up of one or several Cellular V2X driver modules.

7.1.1 Indexing scheme

Users of the Cellular V2X driver identify controller resources using an indexing scheme as describe in the Ethernet driver [5].

[CP_SWS_CV2x_00010]{DRAFT} ⌈The Cellular V2X Driver is using a zero-based index to abstract the access for hardware abstraction layer.⌋*()*

Note: The index CV2xCtrlId within configuration corresponds to the augment CtrlId in APIs defined in chapter 8.3.

[CP_SWS_CV2x_00011]{DRAFT} ⌈A buffer index (BufId) identifies a Cellular V2X buffer processed by Cellular V2X Driver API functions.⌋*()*

[CP_SWS_CV2x_00028]{DRAFT} ⌈Each controller's buffers are identified by buffer indexes 0 to (n-1) where n is the number of buffers processed by the corresponding controller, and it can be configured by CV2xCtrlRxBufTotal and CV2xCtrlTxBufTotal for receiving and transmitting respectively.⌋*()*

[CP_SWS_CV2x_00029]{DRAFT} ⌈Buffer indexes are valid within a tuple <CtrlId, BufId> only.⌋*()*

[CP_SWS_CV2x_00030]{DRAFT} ⌈A BufId uniquely identifies the buffer used for a Cellular V2X Driver.⌋*()*

7.1.2 General requirements

This chapter lists requirements that shall be fulfilled by Cellular V2X Driver module implementations. The Cellular V2X Driver module environment comprises all modules which are calling interfaces of the Cellular V2X Driver module.

[CP_SWS_CV2x_00012]{DRAFT} ⌈The Cellular V2X Driver shall ensure that the base addresses of all reception and transmission buffers fulfill the memory alignment requirements for all AUTOSAR data types of the respective platform such that efficient DMA and Memcopy operations are possible.⌋*()*

[CP_SWS_CV2x_00013]{DRAFT} ⌈The Cellular V2X Driver shall call EthIf_TxConfirmation to indicate a successful transmission from the Interrupt routine (if the notification has been enabled through EthIfTxConfirmationFunction).⌋*()*

[CP_SWS_CV2x_00014]{DRAFT} ⌈The Cellular V2X Driver shall call EthIf_RxIndication to indicate a successful reception from the Interrupt routine.⌋*()*

7.1.3 Per-packet-base parameters

For the Cellular V2X Driver it is important to be able to configure the transmission and the reception parameters for a destined radio of the Cellular V2X.

[CP_SWS_CV2x_00015]{DRAFT} ⌈The Cellular V2X Driver shall provide an API CV2x_GetBufCV2xPC5RxParams that provide a sequence of buffer-based reception parameters related to a received packet.⌋*(CP_SRS_CnV2X_00301)*

[CP_SWS_CV2x_00017]{DRAFT} ⌈The Cellular V2X Driver shall provide an API CV2x_GetBufCV2xPC5TxParams that provide a sequence of buffer-based transmission parameters related to a transmitted packet.⌋*(CP_SRS_CnV2X_00301)*

[CP_SWS_CV2x_00019]{DRAFT} ⌈The Cellular V2X Driver shall provide an API CV2x_SetBufCV2xPC5TxParams that sets a sequence of buffer-based transmission parameters related to a transmitted packet.⌋*(CP_SRS_CnV2X_00301)*

7.1.4 Key/Value parameter mapping

[CP_SWS_CV2x_00021]{DRAFT} ⌈For unique reference to transmission and reception parameters of a sent or received Cellular V2X packet respectively, unique enumeration values shall be used within this module.⌋*(CP_SRS_CnV2X_00301)*

[CP_SWS_CV2x_00023]{DRAFT} ⌈API CV2x_GetBufCV2xPC5RxParams using the type CV2x_BufCV2xPC5RxParamIdType shall convert the following parameters defined in [1] to uint32 or uint8 type.⌋*(CP_SRS_CnV2X_00301)*

[CP_SWS_CV2x_00024]{DRAFT} ⌈

ParamId	*ParamValue Type*
CV2X_BUFCV2XPC5RXPID_SRC_LAYER2_ID	uint32
CV2X_BUFCV2XPC5RXPID_DST_LAYER2_ID	uint32
CV2X_BUFCV2XPC5RXPID_PPPP	uint8
CV2X_BUFCV2XPC5RXPID_CBR	uint8
CV2X_BUFCV2XPC5RXPID_MAX_DATA_RATE	uint32
CV2X_BUFCV2XPC5RXPID_TRANSACTION_ID_32	uint32

⌋*(CP_SRS_CnV2X_00301)*

[CP_SWS_CV2x_00025]{DRAFT} ⌈API CV2x_GetBufCV2xPC5TxParams and API CV2x_SetBufCV2xPC5TxParams using the CV2x_BufCV2xPC5TxParamIdType shall convert the following parameters defined in [1] to uint32 or uint8 type.⌋*(CP_SRS_CnV2X_00301)*

[CP_SWS_CV2x_00026]{DRAFT} ⌈

ParamId	*ParamValue Type*
CV2X_BUFCV2XPC5TXPID_PDCP_SDU_TYPE	uint8
CV2X_BUFCV2XPC5TXPID_SRC_LAYER2_ID	uint32
CV2X_BUFCV2XPC5TXPID_DST_LAYER2_ID	uint32
CV2X_BUFCV2XPC5TXPID_PPPP	uint8
CV2X_BUFCV2XPC5TXPID_CBR	uint8
CV2X_BUFCV2XPC5TXPID_TRAFFIC_PERIOD	uint32
CV2X_BUFCV2XPC5TXPID_SRC_IP_ADDR	uint32
CV2X_BUFCV2XPC5TXPID_TRANSACTION_ID_32	uint32

⌋*(CP_SRS_CnV2X_00301)*

[CP_SWS_CV2x_00027]{DRAFT} ⌈API CV2x_GetChanTxParamIdType using the CV2x_GetChanTxParamIdType shall convert the following parameters defined in [1] to uint32 type.⌋*(CP_SRS_CnV2X_00301)*

[CP_SWS_CV2x_00031]{DRAFT} ⌈

ParamId	*ParamValue Type*
CV2X_GETCHRXPID_CBR	uint32
CV2X_GETCHRXPID_TP	uint32
CV2X_GETCHRXPID_SYNC_TYPE	uint32
CV2X_GETCHRXPID_SYNC_STATUS	uint32

⌋*(CP_SRS_CnV2X_00301)*

7.2 Error Classification

This chapter lists and classifies all errors that can be detected within this software module. Each error is classified according to relevance (development / production) and related error code. For development errors, a value is defined.

7.2.1 Development Errors

[SWS_CV2x_00126]{DRAFT} **Definiton of development errors in module CV2x** ⌈

Type of error	*Related error code*	*Error value*
Invalid controller index **Tags:** atp.Status=draft	CV2X_E_INV_CTRL_IDX	0x01
CV2x module was not initialized **Tags:** atp.Status=draft	CV2X_E_UNINIT	0x02
Invalid pointer in parameter list **Tags:** atp.Status=draft	CV2X_E_PARAM_POINTER	0x03
Invalid parameter **Tags:** atp.Status=draft	CV2X_E_INV_PARAM	0x04
Invalid mode **Tags:** atp.Status=draft	CV2X_E_INV_MODE	0x05

⌋()

7.2.2 Runtime Errors

There are no runtime errors.

7.2.3 Transient Faults

There are no runtime errors.

7.2.4 Production Errors

There are no runtime errors.

7.2.5 Extended Production Errors

There are no extended production errors. Extended production errors are handled as events of the Diagnostic Event Manager. The event IDs are defined in the following

tables, while the actual values are assigned externally by the configuration of the Diagnostic Event Manager, and are included in the module via Dem.h.

[CP_SWS_CV2x_00502]{DRAFT} ⌈

Error Name:	CV2X_E_ACCESS	
Short Description:	Cellular V2X controller access failure	
Long Description:	Monitors the access the Cellular V2X controller	
Detection Criteria:	Fail	When access to the Cellular V2X controller fails, the module shall report the extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.
	Pass	When access to the Cellular V2X controller succeeds, the module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.
Secondary Parameters:	None	
Time Required:	None	
Monitor Frequency:	None	

⌋()

8 API specification

8.1 Imported types

In this chapter all types included from the following files are listed.

[CP_SWS_CV2x_01001] Definition of imported datatypes of module CV2x ⌈

Module	*Header File*	*Imported Type*
ComStack_Types	ComStack_Types.h	BufReq_ReturnType
	ComStackTypes.h	TimeStampQualType (draft)
	ComStackTypes.h	TimeStampType (draft)
	ComStackTypes.h	TimeTupleType (draft)
Dem	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
Eth	Eth_GeneralTypes.h	Eth_BufIdxType
	Eth_GeneralTypes.h	Eth_DataType
	Eth_GeneralTypes.h	Eth_FrameType
	Eth_GeneralTypes.h	Eth_ModeType
	Eth_GeneralTypes.h	Eth_RxStatusType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

⌋()

8.2 Type definitions

8.2.1 CV2x_ConfigType

[CP_SWS_CV2x_01002]{DRAFT} **Definition of datatype CV2x_ConfigType** ⌈

Name	CV2x_ConfigType (draft)
Kind	Structure
Description	Implementation specific structure of the post build configuration **Tags:** atp.Status=draft
Available via	CV2x.h

⌋()

8.2.2 CV2x_StateType

[CP_SWS_CV2x_01003]{DRAFT} **Definition of datatype CV2x_StateType** ⌈

Name	CV2x_StateType (draft)		
Kind	Enumeration		
Range	CV2X_STATE_UNINIT	0x00	Driver is not yet configured
	CV2X_STATE_INIT	0x01	Driver is configured
Description	Wireless parameters for a packet that has been received. **Tags:** atp.Status=draft		
Available via	CV2x_GeneralTypes.h		

⌋()

8.2.3 CV2x_BufCV2xPC5RxParamIdType

[CP_SWS_CV2x_01004]{DRAFT} **Definition of datatype CV2x_BufCV2xPC5Rx ParamIdType** ⌈

Name	CV2x_BufCV2xPC5RxParamIdType (draft)		
Kind	Enumeration		
Range	CV2X_ BUFCV2XPC5RXPID_ SRC_LAYER2_ID	0x00	Source Layer 2 ID of Ceulluar V2X packet
	CV2X_ BUFCV2XPC5RXPID_ DST_LAYER2_ID	0x01	Destination Layer 2 ID of Cellular V2X packet
	CV2X_ BUFCV2XPC5RXPID_ PPPP	0x02	ProSe per-packet priority
	CV2X_ BUFCV2XPC5RXPID_CBR	0x03	Channel busy rate
	CV2X_ BUFCV2XPC5RXPID_ MAX_DATA_RATE	0x04	Max data rate
	CV2X_ BUFCV2XPC5RXPID_ TRANSACTION_ID_32	0x05	Unique id of a frame that has been received
Description	Wireless parameters for a packet that has been received. **Tags:** atp.Status=draft		
Available via	CV2x_GeneralTypes.h		

⌋()

8.2.4 CV2x_BufCV2xPC5TxParamIdType

[CP_SWS_CV2x_01009]{DRAFT} **Definition of datatype CV2x_BufCV2xPC5Tx ParamIdType** ⌈

Name	CV2x_BufCV2xPC5TxParamIdType (draft)		
Kind	Enumeration		
Range	CV2X_ BUFCV2XPC5TXPID_ PDCP_SDU_TYPE	0x00	Network layer protocol type.
	CV2X_ BUFCV2XPC5TXPID_ SRC_LAYER2_ID	0x01	Source Layer 2 ID of Ceulluar V2X packet
	CV2X_ BUFCV2XPC5TXPID_DST_ LAYER2_ID	0x02	Destination Layer 2 ID of Cellular V2X packet
	CV2X_ BUFCV2XPC5TXPID_ PPPP	0x03	ProSe per-packet priority
	CV2X_ BUFCV2XPC5TXPID_PDB	0x04	Packet Delay Budget
	CV2X_ BUFCV2XPC5TXPID_ TRAFFIC_PERIOD	0x05	Traffic Period
	CV2X_ BUFCV2XPC5TXPID_ SRC_IP_ADDR	0x06	Soruce IP address
	CV2X_ BUFCV2XPC5TXPID_ TRANSACTION_ID_16	0x07	Unique id of a frame to be transmitted
Description	Wireless parameters for a packet that has to be transmitted. **Tags:** atp.Status=draft		
Available via	CV2x_GeneralTypes.h		

⌋()

8.2.5 CV2x_GetChanTxParamIdType

[CP_SWS_CV2x_01005]{DRAFT} **Definition of datatype CV2x_GetChanTxParam IdType** ⌈

Name	CV2x_GetChanTxParamIdType (draft)		
Kind	Enumeration		
Range	CV2X_GETCHTXPID_CBR	0x00	Channel Busy Ratio
	CV2X_GETCHTXPID_TP	0x01	Transmit Power
	CV2X_GETCHTXPID_ SYNC_TYPE	0x02	Source of Synchornizaiton
	CV2X_GETCHTXPID_ SYNC_STATUS	0x03	Status of Sychonization

▽

△

Description	Wireless Channel parameters acquire for receive side. **Tags:** atp.Status=draft
Available via	CV2x_GeneralTypes.h

⌋()

8.3 Function definitions

8.3.1 CV2x_Init

[CP_SWS_CV2x_01010]{DRAFT} **Definition of API function CV2x_Init** ⌈

Service Name	CV2x_Init (draft)	

Syntax	`void CV2x_Init (`
	`  const CV2x_ConfigType* CfgPtr`
	`)`	

Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CfgPtr	Points to the implementation specific structure
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Initialize the Cellular V2X driver **Tags:** atp.Status=draft	
Available via	CV2x.h	

⌋()

[CP_SWS_CV2x_01011]{DRAFT} ⌈The function CV2x_Init shall store the access address to the configuration structure CV2xConfigSet for subsequent API calls.⌋()

[CP_SWS_CV2x_01012]{DRAFT} ⌈The function CV2x_Init shall for initialize all configured Cellular V2X controllers in the current CV2xConfigSet, operations can include:

- Disable all controller
- Clear pending Cellular V2X interrupts
- Configure all controller configuration parameters (e.g. frame length, ...)
- Configure all transmit / receive resources (e.g. buffer initialization)
- Delete all pending transmit and receive requests

⌋()

[CP_SWS_CV2x_01013]{DRAFT} ⌈The function CV2x_Init shall set the state of the component to CV2X_STATE_INIT when all initialization operations complete.⌋()

[CP_SWS_CV2x_01014]{DRAFT} ⌈The function CV2x_Init shall check the access to the Cellular V2X controller. If the check fails, the function CV2x_Init shall raise the production error CV2X_E_ACCESS⌋*()*

8.3.2 CV2x_GetVersionInfo

[CP_SWS_CV2x_01016]{DRAFT} **Definition of API function CV2x_GetVersionInfo** ⌈

Service Name	CV2x_GetVersionInfo (draft)	

Syntax	`void CV2x_GetVersionInfo (`
	`  Std_VersionInfoType* VersionInfoPtr`
	`)`	

Service ID [hex]	0x02	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	VersionInfoPtr	Pointer to where to store the version information of this module.
Return value	None	
Description	Returns the version information of this module. **Tags:** atp.Status=draft	
Available via	CV2x.h	

⌋*()*

[CP_SWS_CV2x_01017]{DRAFT} ⌈If development error detection is enabled: the function CV2x_GetVersionInfo shall check the parameter VersionInfoPtr for being valid. If the check fails, the function CV2x_GetVersionInfo shall raise the development error CV2X_E_PARAM_POINTER.⌋*()*

8.3.3 CV2x_SetControllerMode

[CP_SWS_CV2x_01018]{DRAFT} **Definition of API function CV2x_SetControllerMode** ⌈

Service Name	CV2x_SetControllerMode (draft)	

Syntax	`Std_ReturnType CV2x_SetControllerMode (`
	`  uint8 CtrlId,`
	`  Eth_ModeType CtrlMode`
	`)`	

Service ID [hex]	0x03	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlId	Index of the controller within the context of the Cellular V2X Driver

▽

△

	CtrlMode	ETH_MODE_DOWN: disable the controller ETH_MODE_ ACTIVE: enable the controller
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: controller mode could not be changed
Description	Enables / disables the indexed controller. **Tags:** atp.Status=draft	
Available via	CV2x.h	

⌋()

[CP_SWS_CV2x_01019]{DRAFT} ⌈The function CV2x_SetControllerMode shall put the controller in the specified mode given in the parameter 'CtrlMode':

- Upon mode ETH_MODE_DOWN the driver shall: Disable the Cellular V2X controller;Reset all transmit and receive buffers (i.e. ignore all pending transmission and reception requests)
- Upon mode ETH_MODE_ACTIVE, the driver shall: Enable all transmit and receive buffers;Enable the Cellular V2X controller

⌋()

[CP_SWS_CV2x_01020]{DRAFT} ⌈If development error detection is enabled: the function CV2x_SetControllerMode shall check that the service CV2x_Init was previously called. If the check fails, the function CV2x_SetControllerMode shall raise the development error CV2X_E_UNINIT otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[CP_SWS_CV2x_01021]{DRAFT} ⌈If development error detection is enabled: the function CV2x_SetControllerMode shall check the parameter CtrlId for being valid. If the check fails, the function CV2x_SetControllerMode shall raise the development error CV2X_E_INV_CTRL_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋()

[CP_SWS_CV2x_01022]{DRAFT} ⌈If development error detection is enabled: the function CV2x_SetControllerMode shall check that the service CV2x_Init was previously called. If the check fails, the function CV2x_SetControllerMode shall raise the development error CV2X_E_UNINIT otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[CP_SWS_CV2x_01023]{DRAFT} ⌈The function CV2x_SetControllerMode requires CV2x_Init being called first.⌋()

8.3.4 CV2x_GetControllerMode

[CP_SWS_CV2x_01024]{DRAFT} **Definition of API function CV2x_GetController Mode** ⌈

Service Name	CV2x_GetControllerMode (draft)	
Syntax	`Std_ReturnType CV2x_GetControllerMode ( uint8 CtrlId, Eth_ModeType* CtrlModePtr )`	
Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlId	Index of the controller within the context of the Cellular V2X Driver
Parameters (inout)	None	
Parameters (out)	CtrlModePtr	ETH_MODE_DOWN: disable the controller ETH_MODE_ ACTIVE: enable the controller
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: controller mode could not be changed
Description	Obtains the state of the indexed controller. **Tags:** atp.Status=draft	
Available via	CV2x.h	

⌋()

[CP_SWS_CV2x_01025]{DRAFT} ⌈The function CV2x_GetControllerMode shall read the current controller mode.⌋()

[CP_SWS_CV2x_01026]{DRAFT} ⌈If development error detection is enabled: the function CV2x_GetControllerMode shall check that the service Cv2x_Init was previously called. If the check fails, the function CV2x_GetControllerMode shall raise the development error CV2X_E_UNINIT otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[CP_SWS_CV2x_01027]{DRAFT} ⌈If development error detection is enabled: the function CV2x_GetControllerMode shall check the parameter CtrlId for being valid. If the check fails, the function CV2x_GetControllerMode shall raise the development error CV2X_E_INV_CTRL_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋()

[CP_SWS_CV2x_01028]{DRAFT} ⌈If development error detection is enabled: the function CV2x_GetControllerMode shall check the parameter CtrlModePtr for being valid. If the check fails, the function CV2x_GetControllerMode shall raise the development error CV2X_E_PARAM_POINTER otherwise (if DET is disabled) return E_NOT_OK.⌋()

[CP_SWS_CV2x_01029]{DRAFT} ⌈The function CV2x_GetControllerMode requires CV2x_Init being called first.⌋()

8.3.5 CV2x_ProvideTxBuffer

[CP_SWS_CV2x_01030]{DRAFT} **Definition of API function CV2x_ProvideTx Buffer** ⌈

Service Name	CV2x_ProvideTxBuffer (draft)	
Syntax	`BufReq_ReturnType CV2x_ProvideTxBuffer ( uint8 CtrlId, uint8 Priority, Eth_BufIdxType* BufIdPtr, uint8** BufPtr, uint16* LenBytePtr )`	
Service ID [hex]	0x05	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlId	ndex of the controller within the context of the Cellular V2X Driver
	Priority	Priority value used for selection of different wireless transmit queues
Parameters (inout)	LenBytePtr	In: desired length in bytes, out: granted length in bytes
Parameters (out)	BufIdPtr	Index to the granted buffer resource. To be used for subsequent requests
	BufPtr	Pointer to the granted buffer
Return value	BufReq_ReturnType	`BUFREQ_OK`: success `BUFREQ_E_NOT_OK`: default error detected `BUFREQ_E_BUSY`: all buffers in use `BUFREQ_E_OVFL`: requested buffer too large
Description	Provides access to a transmit buffer of the specified controller **Tags:** atp.Status=draft	
Available via	CV2x.h	

⌋()

[CP_SWS_CV2x_01031]{DRAFT} ⌈The function CV2x_ProvideTxBuffer shall provide a transmit buffer resource.⌋()

[CP_SWS_CV2x_01100]{DRAFT} ⌈The Cellular V2X Driver shall lock the buffer until it receives a subsequent call of CV2x_Transmit service with the buffer index returned in the BufIdxPtr parameter.⌋()

[CP_SWS_CV2x_01032]{DRAFT} ⌈All locked transmit buffers shall be released if the controller is disabled via CV2x_SetControllerMode.⌋()

[CP_SWS_CV2x_01033]{DRAFT} ⌈If a buffer requested with Cv2x_ProvideTxBuffer that is larger than the available buffer length, the buffer shall not be locked but return the available length and BUFREQ_E_OVFL.⌋()

[CP_SWS_CV2x_01034]{DRAFT} ⌈If all available buffers are in use the component shall return BUFREQ_E_BUSY.⌋()

[CP_SWS_CV2x_01035]{DRAFT} ⌈If development error detection is enabled: the function CV2x_ProvideTxBuffer shall check that the service CV2x_Init was previously called. If the check fails, the function CV2x_ProvideTxBuffer shall raise the development error CV2X_E_UNINIT and return BUFREQ_E_NOT_OK.⌋()

[CP_SWS_CV2x_01036]{DRAFT} ⌈If development error detection is enabled: the function CV2x_ProvideTxBuffer shall check the parameter CtrlId for being valid. If the check fails, the function CV2x_ProvideTxBuffer shall raise the development error CV2X_E_INV_CTRL_IDX and return BUFREQ_E_NOT_OK.⌋*()*

[CP_SWS_CV2x_01037]{DRAFT} ⌈If development error detection is enabled: the function CV2x_ProvideTxBuffer shall check the parameter BufIdxPtr for being valid. If the check fails, the function CV2x_ProvideTxBuffer shall raise the development error CV2X_E_PARAM_POINTER and return BUFREQ_E_NOT_OK.⌋*()*

[CP_SWS_CV2x_01038]{DRAFT} ⌈If development error detection is enabled: the function CV2x_ProvideTxBuffer shall check the parameter BufPtr for being valid. If the check fails, the function CV2x_ProvideTxBuffer shall raise the development error CV2X_E_PARAM_POINTER and return BUFREQ_E_NOT_OK.⌋*()*

[CP_SWS_CV2x_01039]{DRAFT} ⌈If development error detection is enabled: the function CV2x_ProvideTxBuffer shall check the parameter LenBytePtr for being valid. If the check fails, the function CV2x_ProvideTxBuffer shall raise the development error CV2X_E_PARAM_POINTER and return BUFREQ_E_NOT_OK.⌋*()*

[CP_SWS_CV2x_01040]{DRAFT} ⌈The function CV2x_ProvideTxBuffer requires requires CV2x_Init being called first..⌋*()*

8.3.6 CV2x_Transmit

[CP_SWS_CV2x_01041]{DRAFT} **Definition of API function CV2x_Transmit** ⌈

Service Name	CV2x_Transmit (draft)	
Syntax	`Std_ReturnType CV2x_Transmit ( uint8 CtrlId, Eth_BufIdxType BufId, boolean TxConfirmation, uint16 LenByte )`	
Service ID [hex]	0x06	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlId	Index of the controller within the context of the Cellular V2X Driver
	BufId	Index of the buffer resource
	TxConfirmation	Activates transmission confirmation
	LenByte	Data length in byte (Adaptation Frame length)
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: transmission failed
Description	Triggers transmission of a previously filled transmit buffer **Tags:** atp.Status=draft	
Available via	CV2x.h	

⌋*()*

[CP_SWS_CV2x_01042]{DRAFT} ⌈The function CV2x_Transmit shall trigger the transmission of a previously filled transmit buffer. After transmission, the driver needs to release the allocated buffer. It is up to the implementation when the actual buffer release shall occur, e.g. within the context of the CV2x_TxConfirmation, the CV2x_MainFunction, or during the next CV2x_ProvideTxBuffer.⌋*()*

[CP_SWS_CV2x_01043]{DRAFT} ⌈All pending transmit buffers shall be released if the controller is disabled via CV2x_SetControllerMode.⌋*()*

[CP_SWS_CV2x_01044]{DRAFT} ⌈If development error detection is enabled: the function CV2x_Transmit shall check that the service CV2x_Init was previously called. If the check fails, the function CV2x_Transmit shall raise the development error CV2X_E_UNINIT otherwise (if DET is disabled) return E_NOT_OK.⌋*()*

[CP_SWS_CV2x_01045]{DRAFT} ⌈If development error detection is enabled: the function CV2x_Transmit shall check the parameter CtrlId for being valid. If the check fails, the function CV2x_Transmit shall raise the development error CV2X_E_INV_CTRL_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋*()*

[CP_SWS_CV2x_01046]{DRAFT} ⌈If development error detection is enabled: the function CV2x_Transmit shall check the parameter BufIdx for being valid. If the check fails, the function CV2x_Transmit shall raise the development error CV2X_E_INV_PARAM otherwise (if DET is disabled) return E_NOT_OK.⌋*()*

[CP_SWS_CV2x_01047]{DRAFT} ⌈If development error detection is enabled: the function CV2x_Transmit shall check the controller mode for being active (ETH_MODE_ACTIVE). If the check fails, the function CV2x_Transmit shall raise the development error CV2X_E_INV_MODE otherwise (if DET is disabled) return E_NOT_OK.⌋*()*

[CP_SWS_CV2x_01048]{DRAFT} ⌈The function CV2x_Transmit requires requires CV2x_ProvideTxBuffer being called first.⌋*()*

8.3.7 CV2x_TxConfirmation

[CP_SWS_CV2x_01049]{DRAFT} **Definition of API function CV2x_TxConfirmation** ⌈

Service Name	CV2x_TxConfirmation (draft)	

Syntax	`void CV2x_TxConfirmation (`
	`  uint8 CtrlId`
	`)`	

Service ID [hex]	0x07	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlId	Index of the controller within the context of the Cellular V2X Driver
Parameters (inout)	None	

▽

△

Parameters (out)	None
Return value	None
Description	Triggers transmission confirmation **Tags:** atp.Status=draft
Available via	CV2x.h

⌋()

[CP_SWS_CV2x_01050]{DRAFT} ⌈The function CV2x_TxConfirmation shall check all filled transmit buffers for successful transmission. The function CV2x_TxConfirmation issues transmit confirmation for each transmitted frame using the callback function EthIf_TxConfirmation if requested by the previous call of CV2x_Transmit service.⌋()

[CP_SWS_CV2x_01051]{DRAFT} ⌈If transmission confirmation was enabled by a previous call to CV2x_Transmit function, the function CV2x_TxConfirmation shall release the buffer resource.⌋()

[CP_SWS_CV2x_01052]{DRAFT} ⌈If development error detection is enabled: the function CV2x_TxConfirmation shall check that the service CV2x_Init was previously called. If the check fails, the function CV2x_TxConfirmation shall raise the development error CV2X_E_UNINIT.⌋()

[CP_SWS_CV2x_01053]{DRAFT} ⌈If development error detection is enabled: the function CV2x_TxConfirmation shall check the parameter CtrlId for being valid. If the check fails, the function CV2x_TxConfirmation shall raise the development error CV2X_E_INV_CTRL_IDX.⌋()

[CP_SWS_CV2x_01054]{DRAFT} ⌈If development error detection is enabled: the function CV2x_TxConfirmation shall check the controller mode for being active (ETH_MODE_ACTIVE). If the check fails, the function CV2x_TxConfirmation shall raise the development error CV2X_E_INV_MODE.⌋()

[CP_SWS_CV2x_01055]{DRAFT} ⌈The function CV2x_TxConfirmation requires requires CV2x_Init being called first.⌋()

8.3.8 CV2x_Receive

[CP_SWS_CV2x_01056]{DRAFT} **Definition of API function CV2x_Receive** ⌈

Service Name	CV2x_Receive (draft)

Syntax	`void CV2x_Receive (`
	`  uint8 CtrlId,`
	`  Eth_RxStatusType* RxStatusPtr`
	`)`

Service ID [hex]	0x08
Sync/Async	Synchronous
Reentrancy	Non Reentrant

▽

△

Parameters (in)	CtrlId	Index of the controller within the context of the Cellular V2X Driver
Parameters (inout)	None	
Parameters (out)	RxStatusPtr	Indicates whether a frame has been received and if so, whether more frames are available or frames got lost.
Return value	None	
Description	Triggers frame reception **Tags:** atp.Status=draft	
Available via	CV2x.h	

⌋()

[CP_SWS_CV2x_01057]{DRAFT} ⌈The function CV2x_Receive shall read the next frame from the receive buffers.⌋()

[CP_SWS_CV2x_01101]{DRAFT} ⌈The function CV2x_Receive passes the received frame to the Ethernet interface using the callback function EthIf_RxIndication and indicates if there are more frames in the receive buffers.⌋()

[CP_SWS_CV2x_01058]{DRAFT} ⌈If development error detection is enabled: the function CV2x_Receive shall check that the service CV2x_Init was previously called. If the check fails, the function CV2x_Receive shall raise the development error CV2X_E_UNINIT.⌋()

[CP_SWS_CV2x_01059]{DRAFT} ⌈If development error detection is enabled: the function CV2x_Receive shall check the parameter CtrlId for being valid. If the check fails, the function CV2x_Receive shall raise the development error CV2X_E_INV_CTRL_IDX.⌋()

[CP_SWS_CV2x_01060]{DRAFT} ⌈If development error detection is enabled: the function CV2x_Receive shall check the controller mode for being active (ETH_MODE_ACTIVE). If the check fails, the function CV2x_Receive shall raise the development error CV2X_E_INV_MODE.⌋()

[CP_SWS_CV2x_01061]{DRAFT} ⌈The received broadcast frames shall be indicated to the Ethernet Interface by the callback function EthIf_RxIndication.⌋()

[CP_SWS_CV2x_01062]{DRAFT} ⌈The function CV2x_Receive requires CV2x_Init being called first.⌋()

8.3.9 CV2x_GetBufCV2xPC5RxParams

[CP_SWS_CV2x_01063]{DRAFT} **Definition of API function CV2x_GetBufCV2x PC5RxParams** ⌈

Service Name	CV2x_GetBufCV2xPC5RxParams (draft)	
Syntax	`Std_ReturnType CV2x_GetBufCV2xPC5RxParams ( uint8 CtrlId, const CV2x_BufCV2xPC5RxParamIdType* RxParamIds, uint32* ParamValues, uint8 NumParams )`	
Service ID [hex]	0x09	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlId	Index of the controller within the context of the Cellular V2X Driver
	RxParamIds	IDs of the Parameter that are requested
	NumParams	Number of Parameters are requested
Parameters (inout)	None	
Parameters (out)	ParamValues	Values of the parameters requested
Return value	Std_ReturnType	`E_OK`: Success `E_NOT_OK`: failed reading parameters
Description	Read out values related to a received packet. For example, this could be CBR to one single packet.This API is valid only within the context of CV2x_Receive **Tags:** atp.Status=draft	
Available via	CV2x.h	

⌋()

[CP_SWS_CV2x_01064]{DRAFT} ⌈If development error detection is enabled: the function CV2x_GetBufCV2xPC5RxParams shall check that CV2x_Init was previously called. If the check fails, the function CV2x_GetBufCV2xPC5RxParams shall raise the development error CV2X_E_UNINIT.⌋(*SRS_BSW_00487*)

[CP_SWS_CV2x_01065]{DRAFT} ⌈If development error detection is enabled: the function CV2x_GetBufCV2xPC5RxParams shall check the parameter CtrlId for being valid. If the check fails, the function CV2x_GetBufCV2xPC5RxParams shall raise the development error CV2X_E_INV_CTRL_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋()

[CP_SWS_CV2x_01066]{DRAFT} ⌈If development error detection is enabled: the function CV2x_GetBufCV2xPC5RxParams shall check the parameter RxParamIds for being valid. If the check fails, the function CV2x_GetBufCV2xPC5RxParams shall raise the development error CV2X_E_PARAM_POINTER.⌋()

[CP_SWS_CV2x_01067]{DRAFT} ⌈If development error detection is enabled: the function CV2x_GetBufCV2xPC5RxParams shall check the parameter ParamValues for being valid. If the check fails, the function CV2x_GetBufCV2xPC5RxParams shall raise the development error CV2X_E_PARAM_POINTER.⌋()

8.3.10 CV2x_GetBufCV2xPC5TxParams

[CP_SWS_CV2x_01069]{DRAFT} **Definition of API function CV2x_GetBufCV2x PC5TxParams** ⌈

Service Name	CV2x_GetBufCV2xPC5TxParams (draft)	
Syntax	`Std_ReturnType CV2x_GetBufCV2xPC5TxParams ( uint8 CtrlId, const CV2x_BufCV2xPC5TxParamIdType* TxParamIds, uint32* ParamValues, uint8 NumParams )`	
Service ID [hex]	0x0A	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlId	Index of the controller within the context of the Cellular V2X Driver
	TxParamIds	IDs of the Parameter that are requested
	NumParams	Number of Parameters are requested
Parameters (inout)	None	
Parameters (out)	ParamValues	Values of the parameters requested
Return value	Std_ReturnType	`E_OK`: Success `E_NOT_OK`: failed reading parameters
Description	Read out values related to the receive direction for a transmitted packet. For example, this could be transaction ID to one single packet. This API is valid only within the context of CV2x_ TxConfirmation **Tags:** atp.Status=draft	
Available via	CV2x.h	

⌋()

[CP_SWS_CV2x_01070]{DRAFT} ⌈If development error detection is enabled: the function CV2x_GetBufCV2xPC5TxParams shall check that the service CV2x_Init was previously called. If the check fails, the function CV2x_GetBufCV2xPC5TxParams shall raise the development error CV2X_E_UNINIT.⌋(*SRS_BSW_00487*)

[CP_SWS_CV2x_01071]{DRAFT} ⌈If development error detection is enabled: the function CV2x_GetBufCV2xPC5TxParams shall check the parameter CtrlId for being valid. If the check fails, the function CV2x_GetBufCV2xPC5TxParams shall raise the development error CV2X_E_INV_CTRL_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋()

[CP_SWS_CV2x_01072]{DRAFT} ⌈If development error detection is enabled: the function CV2x_GetBufCV2xPC5TxParams shall check the parameter TxParamIds for being valid. If the check fails, the function CV2x_GetBufCV2xPC5TxParams shall raise the development error CV2X_E_PARAM_POINTER.⌋()

[CP_SWS_CV2x_01073]{DRAFT} ⌈If development error detection is enabled: the function CV2x_GetBufCV2xPC5TxParams shall check the parameter ParamValues for being valid. If the check fails, the function CV2x_GetBufCV2xPC5TxParams shall raise the development error CV2X_E_PARAM_POINTER.⌋()

8.3.11 CV2x_SetBufCV2xPC5TxParams

[CP_SWS_CV2x_01074]{DRAFT} **Definition of API function CV2x_SetBufCV2x PC5TxParams** ⌈

Service Name	CV2x_SetBufCV2xPC5TxParams (draft)	
Syntax	`Std_ReturnType CV2x_SetBufCV2xPC5TxParams ( uint8 CtrlId, Eth_BufIdxType BufId, const CV2x_BufCV2xPC5TxParamIdType* TxParamIds, uint32* ParamValues, uint8 NumParams )`	
Service ID [hex]	0x0B	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlId	Index of the controller within the context of the Cellular V2X Driver
	BufId	Index of the buffer resource
	TxParamIds	IDs of the Parameter that are requested
	ParamValues	Values of the Parameters that are provided to the transmit radio
	NumParams	Number of Parameters are requested
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Success `E_NOT_OK`: failed reading parameters
Description	Set values related to the transmit direction for a specific buffer (packet to be sent). For example, this can be PPPP belonging to one single packet. **Tags:** atp.Status=draft	
Available via	CV2x.h	

⌋*()*

[CP_SWS_CV2x_01075]{DRAFT} ⌈If development error detection is enabled: the function CV2x_SetBufCV2xPC5TxParams shall check that the service CV2x_Init was previously called. If the check fails, the function CV2x_SetBufCV2xPC5TxParams shall raise the development error CV2X_E_UNINIT⌋*()*

[CP_SWS_CV2x_01076]{DRAFT} ⌈If development error detection is enabled: the function CV2x_SetBufCV2xPC5TxParams shall check the parameter CtrlId for being valid. If the check fails, the function CV2x_SetBufCV2xPC5TxParams shall raise the development error CV2X_E_INV_CTRL_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋*()*

[CP_SWS_CV2x_01077]{DRAFT} ⌈If development error detection is enabled: the function CV2x_SetBufCV2xPC5TxParams shall check the parameter BufId for being valid. If the check fails, the function CV2x_SetBufCV2xPC5TxParams shall raise the development error CV2X_E_INV_PARAM otherwise (if DET is disabled) return E_NOT_OK.⌋*()*

[CP_SWS_CV2x_01078]{DRAFT} ⌈If development error detection is enabled: the function CV2x_SetBufCV2xPC5TxParams shall check the parameter TxParamIds for being valid. If the check fails, the function CV2x_SetBufCV2xPC5TxParams shall raise the development error CV2X_E_PARAM_POINTER.⌋*()*

[CP_SWS_CV2x_01079]{DRAFT} ⌈If development error detection is enabled: the function CV2x_SetBufCV2xPC5TxParams shall check the parameter ParamValues for being valid. If the check fails, the function CV2x_SetBufCV2xPC5TxParams shall raise the development error CV2X_E_PARAM_POINTER.⌋*()*

8.3.12 CV2x_GetChanCV2xPC5TxParams

[CP_SWS_CV2x_01080]{DRAFT} **Definition of API function CV2x_GetChanCV2x PC5TxParams** ⌈

Service Name	CV2x_GetChanCV2xPC5TxParams (draft)	
Syntax	`Std_ReturnType CV2x_GetChanCV2xPC5TxParams ( uint8 CtrlId, uint8 ChannelId, const CV2x_GetChanTxParamIdType* ParamIds, uint32* ParamValues, uint8 NumParams )`	
Service ID [hex]	0x0C	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlId	Index of the controller within the context of the Cellular V2X Driver (Transceiver Id)
	ChannelId	Index of Transceiver's Radio Channel
	ParamIds	IDs of the Parameters to read
	NumParams	Number of parameters to read
Parameters (inout)	None	
Parameters (out)	ParamValues	Value of the requested Parameters
Return value	Std_ReturnType	`E_OK`: Success `E_NOT_OK`: failed reading parameters
Description	Read values related to the receive direction of the channel. For example, this could be a Channel Busy Ratio (CBR) **Tags:** atp.Status=draft	
Available via	CV2x.h	

⌋*()*

[CP_SWS_CV2x_01081]{DRAFT} ⌈The function CV2x_GetChanCV2xPC5TxParams shall use the type mapping form SWS_CV2x_00027 for the ParamIds and ParamValues parmeters.⌋*()*

[CP_SWS_CV2x_01082]{DRAFT} ⌈If development error detection is enabled: the function CV2x_GetChanCV2xPC5TxParams shall check that the service CV2x_Init was previously called. If the check fails, the function CV2x_GetChanCV2xPC5TxParams shall raise the development error CV2X_E_UNINIT .⌋*()*

[CP_SWS_CV2x_01083]{DRAFT} ⌈If development error detection is enabled: the function CV2x_GetChanCV2xPC5TxParams shall check the parameter CtrlId for being valid. If the check fails, the function CV2x_GetChanCV2xPC5TxParams shall raise

the development error CV2X_E_INV_CTRL_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋()

[CP_SWS_CV2x_01084]{DRAFT} ⌈If development error detection is enabled: the function CV2x_GetChanCV2xPC5TxParams shall check the parameter ChannelId for being valid. If the check fails, the function CV2x_GetChanCV2xPC5TxParams shall raise the development error CV2X_E_PARAM_POINTER.⌋()

[CP_SWS_CV2x_01085]{DRAFT} ⌈If development error detection is enabled: the function CV2x_GetChanCV2xPC5TxParams shall check the parameter ParamIds for being valid. If the check fails, the function CV2x_GetChanCV2xPC5TxParams shall raise the development error CV2X_E_PARAM_POINTER.⌋()

[CP_SWS_CV2x_01086]{DRAFT} ⌈If development error detection is enabled: the function CV2x_GetChanCV2xPC5TxParams shall check the parameter ParamsValues for being valid. If the check fails, the function CV2x_GetChanCV2xPC5TxParams shall raise the development error CV2X_E_PARAM_POINTER.⌋()

8.4 Callback notifications

The Cellular V2X Driver does not provide any callback functions.

8.5 Scheduled functions

8.5.1 CV2x_MainFunction

[CP_SWS_CV2x_02001]{DRAFT} **Definition of scheduled function CV2x_Main Function** ⌈

Service Name	CV2x_MainFunction (draft)

Syntax	`void CV2x_MainFunction (`
	`  void`
	`)`

Service ID [hex]	0x10
Description	Support for indirect transmissions (extended frame timing constraints). Used for polling state changes. Calls EthIf_CtrlModeIndication when the controller mode changed. **Tags:** atp.Status=draft
Available via	SchM_CV2x.h

⌋()

[CP_SWS_CV2x_02002]{DRAFT} ⌈The function CV2x_MainFunction is used for polling state changes. EthIf_CtrlModeIndication shall be called when the controller mode changed.⌋()

[CP_SWS_CV2x_02003]{DRAFT} ⌈The function CV2x_MainFunction is used for hardware / software implementation specific execution of cyclic tasks.⌋()

8.6 Expected interfaces

8.6.1 Mandatory interfaces

This chapter defines all external interfaces, which are required to fulfill the core functionality of the module.

[CP_SWS_CV2x_02004] Definition of mandatory interfaces in module CV2x ⌈

API Function	*Header File*	*Description*
Dem_SetEventStatus	Dem.h	Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEventStatus can safely ignore the return value. This API will be available only if ({Dem/Dem ConfigSet/DemEventParameter/DemEvent ReportingType} == STANDARD_REPORTING)
EthIf_CtrlModeIndication	EthIf.h	Called asynchronously when mode has been read out. Triggered by previous <EthDrv>_SetController Mode call. Can directly be called within the trigger functions.
EthIf_RxIndication	EthIf.h	Receive indication of an Ethernet frame which was received by the indexed controller
EthIf_TxConfirmation	EthIf.h	Confirms frame transmission by the indexed controller

⌋()

[CP_SWS_CV2x_02005]{DRAFT} ⌈The Cellular V2X Driver shall ignore the input Parameter FrameType and PhysAddr in the function EthIf_RxIndication, as FrameType is not used in Cellular V2X communication and PhysAddr is obtained by the function EthIf_GetBufCV2xPC5RxParams.⌋()

8.6.2 Optional interfaces

This chapter defines all external interfaces, which are required to fulfill an optional functionality of the module.

[CP_SWS_CV2x_02009] Definition of optional interfaces in module CV2x ⌈

API Function	*Header File*	*Description*
Det_ReportError	Det.h	Service to report development errors.

⌋()

8.6.3 Configurable interfaces

The Cellular V2X Driver does not use configurable interfaces.

9 Sequence diagrams

The Cellular V2X Driver will interact with Ethernet Interface in the same way as the Ethernet Driver, see sequence diagrams in [6].

10 Configuration specification

Chapter 10.1 specifies the structure (containers) and the parameters of the CV2x module.

Chapter 10.2 specifies additionally published information of the CV2x module.

10.1 Containers and configuration parameters

The following chapters summarize all configuration parameters.

[CP_SWS_CV2x_03001]{DRAFT} ⌈The Cellular V2X Driver module shall reject configurations with partition mappings, which are not supported by the implementation.⌋ *()*

10.1.1 Variant

[CP_SWS_CV2x_03002]{DRAFT} ⌈The Cellular V2X Driver module shall support precompile time, link time and post-build time configuration.⌋ *()*

10.1.2 CV2x

SWS Item	[ECUC_CV2x_00001]
Module Name	CV2x
Description	Configuration of the CV2x module (Cellular V2X Driver).
Post-Build Variant Support	true
Supported Config Variants	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
CV2xConfigSet	1	This container contains the configuration parameters and sub containers of the AUTOSAR CV2x module. **Tags:** atp.Status=draft
CV2xGeneral	1	General Configuration of Cellular V2X Driver **Tags:** atp.Status=draft

10.1.3 CV2xGeneral

SWS Item	[ECUC_CV2x_00002]		
Container Name	CV2xGeneral		
Parent Container	CV2x		
Description	General Configuration of Cellular V2X Driver **Tags:** atp.Status=draft		
Configuration Parameters			

SWS Item	[ECUC_CV2x_00004]		
Parameter Name	CV2xDevErrorDetect		
Parent Container	CV2xGeneral		
Description	Switches the Default Error Tracer (Det) detection and notification ON or OFF. - true: detection and notification is enabled. - false: detection and notification is disabled. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_CV2x_00005]		
Parameter Name	CV2xIndex		
Parent Container	CV2xGeneral		
Description	Specifies the InstanceId of this module instance. If only one instance ispresent, it shall have the Id 0. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_CV2x_00006]
Parameter Name	CV2xMainFunctionPeriod
Parent Container	CV2xGeneral
Description	Specifies the period of main function CV2x_MainFunction in seconds. Cellular V2X driver does not require this information but the BSW scheduler. **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucFloatParamDef

△

Range	]0 .. INF[		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_CV2x_00007]**		
Parameter Name	CV2xVersionInfoApi		
Parent Container	CV2xGeneral		
Description	Enables / Disables version info API. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_CV2x_00008]**		
Parameter Name	CV2xEcucPartitionRef		
Parent Container	CV2xGeneral		
Description	Maps the Cellular V2X driver to zero or multiple ECUC partitions to make the modules API available in this partition. The Cellular V2X driver will operate as an independent instance in each of the partitions. **Tags:** atp.Status=draft		
Multiplicity	0..*		
Type	Reference to EcucPartition		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

[CP_SWS_CV2x_CONSTR_00241]{DRAFT} ⌈The module will operate as an independent instance in each of the partitions, means the called API will only target the partition it is called in.⌋*()*

[CP_SWS_CV2x_CONSTR_00242]{DRAFT} ⌈If CV2xEcucPartitionRef references one or more ECUC partitions, CV2xCtrlEcucPartitionRef shall have a multiplicity of one and reference one of these ECUC partitions as well.⌋*()*

10.1.4 CV2xConfigSet

SWS Item	[ECUC_CV2x_00003]
Container Name	CV2xConfigSet
Parent Container	CV2x
Description	This container contains the configuration parameters and sub containers of the AUTOSAR CV2x module. **Tags:** atp.Status=draft
Configuration Parameters	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
CV2xCtrlConfig	1	Configuration of individual controller **Tags:** atp.Status=draft

10.1.5 CV2xCtrlConfig

SWS Item	[ECUC_CV2x_00009]
Container Name	CV2xCtrlConfig
Parent Container	CV2xConfigSet
Description	Configuration of individual controller **Tags:** atp.Status=draft
Configuration Parameters	

SWS Item	[ECUC_CV2x_00010]		
Parameter Name	CV2xCtrlId		
Parent Container	CV2xCtrlConfig		
Description	Specifies the instance ID of the configured controller. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

SWS Item	[ECUC_CV2x_00011]		
Parameter Name	CV2xCtrlRxBufLenByte		
Parent Container	CV2xCtrlConfig		
Description	Limits the maximum receive buffer length (frame length) in bytes. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 131071		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	[ECUC_CV2x_00013]		
Parameter Name	CV2xCtrlRxBufTotal		
Parent Container	CV2xCtrlConfig		
Description	Configures the number of receive buffers. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	[ECUC_CV2x_00012]		
Parameter Name	CV2xCtrlTxBufLenByte		
Parent Container	CV2xCtrlConfig		
Description	Limits the maximum transmit buffer length (frame length) in bytes. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 131071		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	[ECUC_CV2x_00014]		
Parameter Name	CV2xCtrlTxBufTotal		
Parent Container	CV2xCtrlConfig		
Description	Configures the number of transmit buffers. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	[ECUC_CV2x_00015]		
Parameter Name	CV2xCtrlEcucPartitionRef		
Parent Container	CV2xCtrlConfig		
Description	Maps the Cellular V2X controller to zero or one ECUC partitions. The ECUC partition referenced is a subset of the ECUC partitions where the Cellular V2X driver is mapped to. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Reference to EcucPartition		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
CV2xDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references. **Tags:** atp.Status=draft

10.1.6 CV2xDemEventParameterRefs

SWS Item	**[ECUC_CV2x_00016]**		
Container Name	CV2xDemEventParameterRefs		
Parent Container	CV2xCtrlConfig		
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The Event Id is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references. **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

SWS Item	**[ECUC_CV2x_00017]**		
Parameter Name	CV2X_E_ACCESS		
Parent Container	CV2xDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Controller access failed" has occured. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

A Not applicable requirements

None

B Change history of AUTOSAR traceable items

Please note that the lists in this chapter also include traceable items that have been removed from the specification in a later version. These items do not appear as hyperlinks in the document.

B.1 Traceable item history of this document according to AUTOSAR Release R23-11

B.1.1 Added Specification Items in R23-11

none

B.1.2 Changed Specification Items in R23-11

none

B.1.3 Deleted Specification Items in R23-11

none

B.1.4 Added Constraints in R23-11

none

B.1.5 Changed Constraints in R23-11

none

B.1.6 Deleted Constraints in R23-11

none