CddFiat

Technical Reference

Complex Device Driver

Version 1.06.00

Authors	Jochen Vorreiter
Status	Released

Document Information

History

Author	Date	Version	Remarks
virlg	2010-10-06	1.00.00	First Version
virlg	2010-12-23	1.01.00	Adding content for Class B ECU features
virlg	2011-03-07	1.02.00	Changing the interface for the used IPDU callouts
virlg	2011-05-18	1.02.01	Chapter 6.3 added to describe additional unhandled signals
virlg	2011-08-30	1.03.00	Class C supports EOL. Integration chapter updated
visvjn	2012-11-15	1.04.00	Adding APIs for support of Class C with Wakeup
visvjn	2013-03-12	1.05.00	Adding support for Multichannel ECUs
visvjn	2013-10-15	1.06.00	ESCAN00070746: Adding API CddFiat_StatusC_MessageIndication ESCAN00069050: Adding support for OSEK_NM ESCAN00069060: Adding description about configuration of Multiple Ecu

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_DET.pdf	V2.2.1
[2]	AUTOSAR	AUTOSAR_BasicSoftwareModules.pdf	V1.3.0
[3]	Vector	AN-ISC-8-1124 FGA Class C Nm in MICROSAR	V1.2
[4]	Fiat	07320: Network Operational Specifications for the Class B Network	Rev. 5
[5]	Fiat	07321: Network Operational Specifications for the Class C Network	Rev. 4
[6]	Vector	Technical Reference Fiat Class B Network Management	1.02.00
[7]	Vector	Technical Reference Fiat Class C Network Management	1.02.00

Scope of the Document

This technical reference describes the usage of the CddFiat component.

Contents

Illustrations

Tables

1 Component History

The component history gives an overview over the important milestones that are supported in the different versions of the component.

Component Version	New Features
1.00.00	Initial Version of the component
1.01.00	The component now support Fiat Class B ECU
1.04.00	The component now support Class C wakeup handling
1.05.00	The component now support Multichannel ECUs
1.06.00	The component now support OSEK_NM, Multiple Configuration and Multiple ECU

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module CddFiat.

This module has to be integrated for all FGA AUTOSAR Stack projects including Carry-Back ECUs.

Supported AUTOSAR Release:	3.2	
Supported Configuration Variants:	pre-compile	
Vendor ID:	CDDFIAT_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	CDDFIAT_MODULE_ID	255 decimal (according to ref. [2])

The purpose of this component is to provide a common interface for (Fiat specific) Network Management modules to application software. Furthermore it updates data of Nm related signals of status messages.

2.1 Architecture Overview

The CddFiat is a complex driver implementing FGA specific requirements for the handling of the node status messages.

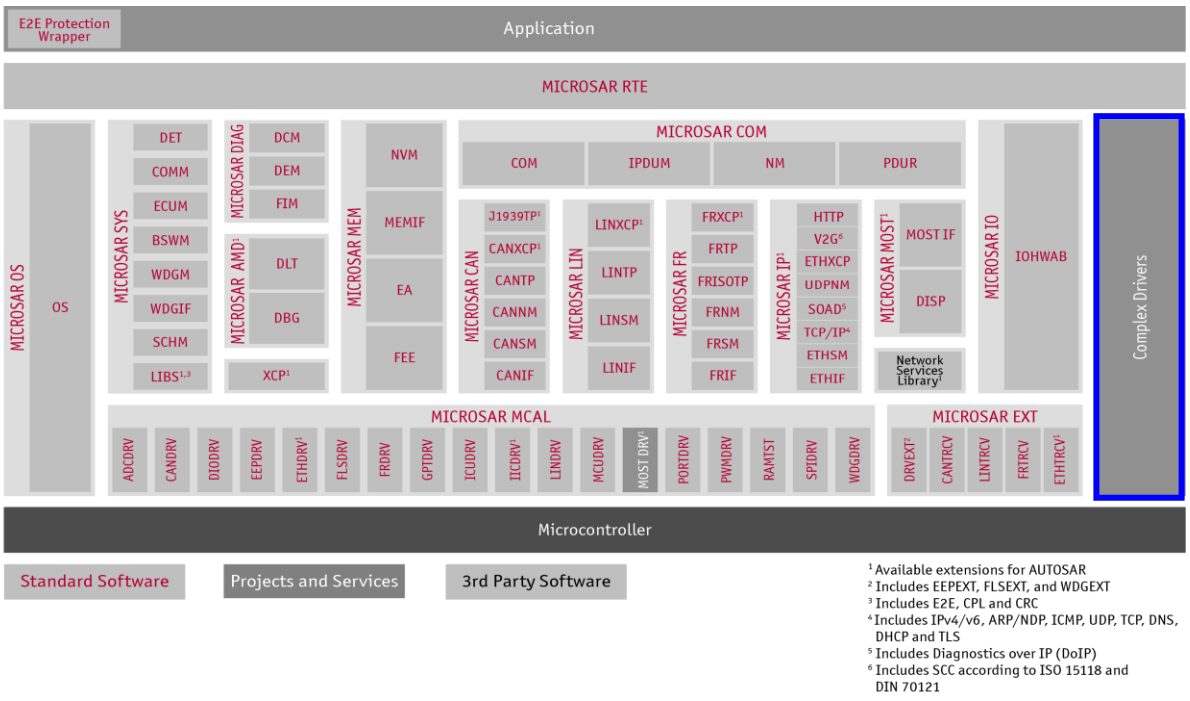

Figure 2-1 AUTOSAR architecture

The next figure shows the interfaces to adjacent modules of the CddFiat. These interfaces are described in chapter 5.

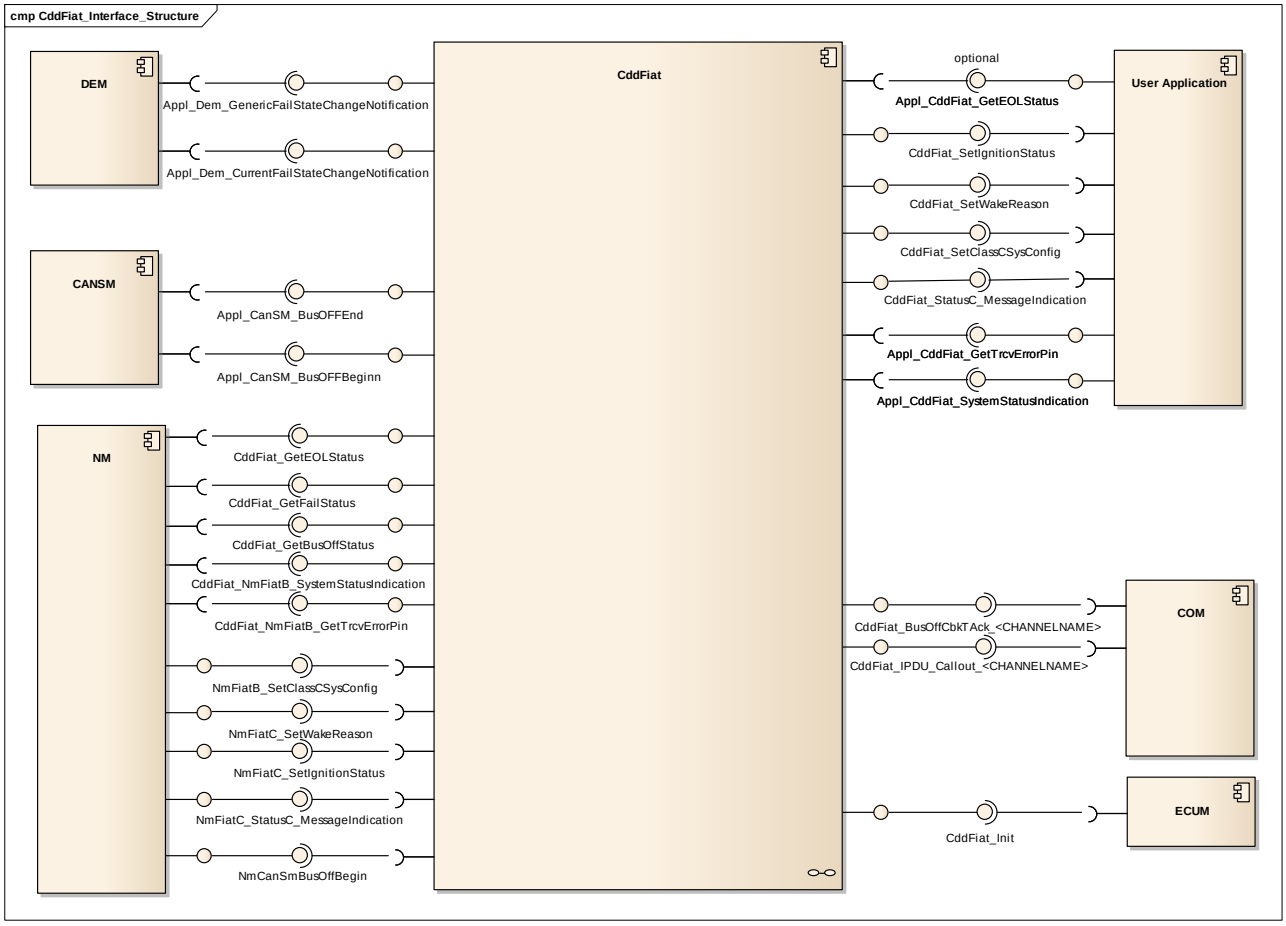

Figure 2-2 Interfaces to adjacent modules of the CddFiat

3 Functional Description

3.1 Features

The features listed in this chapter cover the complete functionality.

The “supported” and “not supported” features are presented in the following two tables.

The following features described are supported:

Supported Feature
Support channels with the NM types Class C non Gateway (with and without wakeup handling), Class B NM, AUTOSAR NM and Osek NM.
Update Generic and Current Dem fail state signal in status message.
Update BusOff state signal in status message.
Update EOL signal in status message.
Support Multiple Configuration and Multiple Ecu.

Table 3-1 Supported features

The following features are not supported:

Not Supported Feature
Class C Gateway Monitoring Functions

Table 3-2 Not supported features

Note
Gateway Monitoring Functions has to be implemented by application as described in [3].

3.2 System Status Indication

The following callback is called by CddFiat for Class B Master channels to indicate the current monitored system configuration to application software:

`void` **`Appl_CddFiat_SystemStatusIndication`**`()` (see 5.3.2)

The passed parameter points to a structure of NmFiatB_SystemConfigType which contains the following fields:

Struct Element Name	C-Type	Description
SystemConfig	uint8*	Pointer to an uint8 array of size arraySize that contains the current System Configuration.
SystemEolConfig	uint8*	Pointer to an uint8 array of size arraySize that contains the current System End Of Line status.
SystemFailConfig	uint8*	Pointer to an uint8 array of size arraySize that contains the current System Fail status.
SystemCurFailConfig	uint8*	Pointer to an uint8 array of size arraySize that contains the current System Current Fail status.
SystemPhysStatus	uint8*	Pointer to an uint8 array of size arraySize that contains the current CAN Bus Wire Failure status.
SystemActiveLoads	boolean*	Pointer to a boolean variable which indicates the current System Active Loads status: ■ TRUE: At least one Slave node has sent its Nm message with AL=1 ■ FALSE: No Slave node has sent its Nm message with AL=1

Table 3-3 NmFiatB_SystemConfigType

Caution
Note that the data of the system configurations is only valid within the call context of Appl_CddFiat_SystemStatusIndication. If the data is needed for later access, it has to be copied to own data structures within the callback function.

Additional information can also be found in technical reference of NmFiatB [6].

3.3 Error Handling

3.3.1 Development Error Reporting

Development errors are reported to Det using the service Det_ReportError() (specified in [1]), if development error reporting is enabled (i.e. pre-compile parameter CDDFIAT_DEV_ERROR_DETECT == STD_ON).

If another module is used for development error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service Det_ReportError().

The reported CddFiat module Id is 255 with the Instance Id 0.

Service ID	Service
CDDFIAT_SID_INIT = 0x01	CddFiat_Init
CDDFIAT_SID_BUSOFF_CBK_T_ACK = 0x02	CddFiat_BusOffCbkTAck
CDDFIAT_SID_IPDU_CALLOUT = 0x03	CddFiat_IPDU_Callout
CDDFIAT_SID_BUSOFF_BEGINN = 0x04	Appl_CanSM_BusOffBegin

Service ID	Service
CDDFIAT_SID_BUSOFF_END = 0x05	Appl_CanSM_BusOffEnd
CDDFIAT_SID_GENERIC_DEM_FAIL_STATE_NOTIFICATION = 0x06	Appl_Dem_GenericFailStateChangeNotification
CDDFIAT_SID_CURRENT_DEM_FAIL_STATE_NOTIFICATION = 0x07	Appl_Dem_CurrentFailStateChangeNotification
CDDFIAT_SID_GET_BUS_OFF_STATUS = 0x08	CddFiat_GetBusOffStatus
CDDFIAT_SID_GET_EOL_STATUS = 0x09	CddFiat_GetEOLStatus
CDDFIAT_SID_GET_FAIL_STATUS = 0x0A	CddFiat_GetFailStatus

Table 3-4 Mapping of service IDs to services

The errors reported to Det are described in the following table:

Error Code		Description
[0x01]	CDDFIAT_E_UNINIT	Component is not initialized
[0x02]	CDDFIAT_E_PARAM	Parameter out of range.
[0x03]	CDDFIAT_E_INVALID_CHANNEL	Invalid channel parameter

Table 3-5 Errors reported to Det

3.3.1.1 Parameter Checking

The following table shows which parameter checks are performed on which services:

Service \ Check	CDDFIAT_E_PARAM	CDDFIAT_E_UNINIT	CDDFIAT_E_INVALID_CHANNEL
CddFiat_Init	■		
CddFiat_BusOffCbkTAck		■	
CddFiat_IPDU_Callout	■	■	
Appl_CanSM_BusOffBegin		■	
Appl_CanSM_BusOffEnd		■	
Appl_Dem_GenericFailStateChangeNotification	■	■	

Service \ Check	CDDFIAT_E_PARAM	CDDFIAT_E_UNINIT	CDDFIAT_E_INVALID_CHANNEL
Appl_Dem_CurrentFailStateChangeNotification	■	■	
CddFiat_GetBusOffStatus	■	■	■
CddFiat_GetEOLStatus	■	■	
CddFiat_GetFailStatus	■	■	

Table 3-6 Development Error Reporting: Assignment of checks to services

AUTOSAR requires that API functions check the validity of their parameters. The checks in Table 3-6 are internal parameter checks of the API functions. These checks are for development error reporting and can be en-/disabled separately. En-/disabling of all checks are an addition to the AUTOSAR standard which requires to en-/disable the complete parameter checking via the parameter `CDDFIAT_DEV_ERROR_DETECT`.

4 Integration

This chapter gives necessary information for the integration of the CddFiat component into an application environment of an ECU.

4.1 Scope of Delivery

CddFiat Configuration File delivered as a template.

4.1.1 Static Files

File Name	Source Code Delivery	Object Code Delivery	Description
CddFiat.c	■		This file implements the CddFiat functionality
CddFiat.h	■		This is the header file of the component

Table 4-1 Static files

4.1.2 Dynamic Files

File Name	Source Code Delivery	Object Code Delivery	Description
CddFiat_Cfg.c	■		This file implements the generated callout for IPDU transmission and the transmit callback after successful transmission of a configured signal.
CddFiat_Cfg.h	■		This file contains the configuration part of this module.

4.2 Include Structure

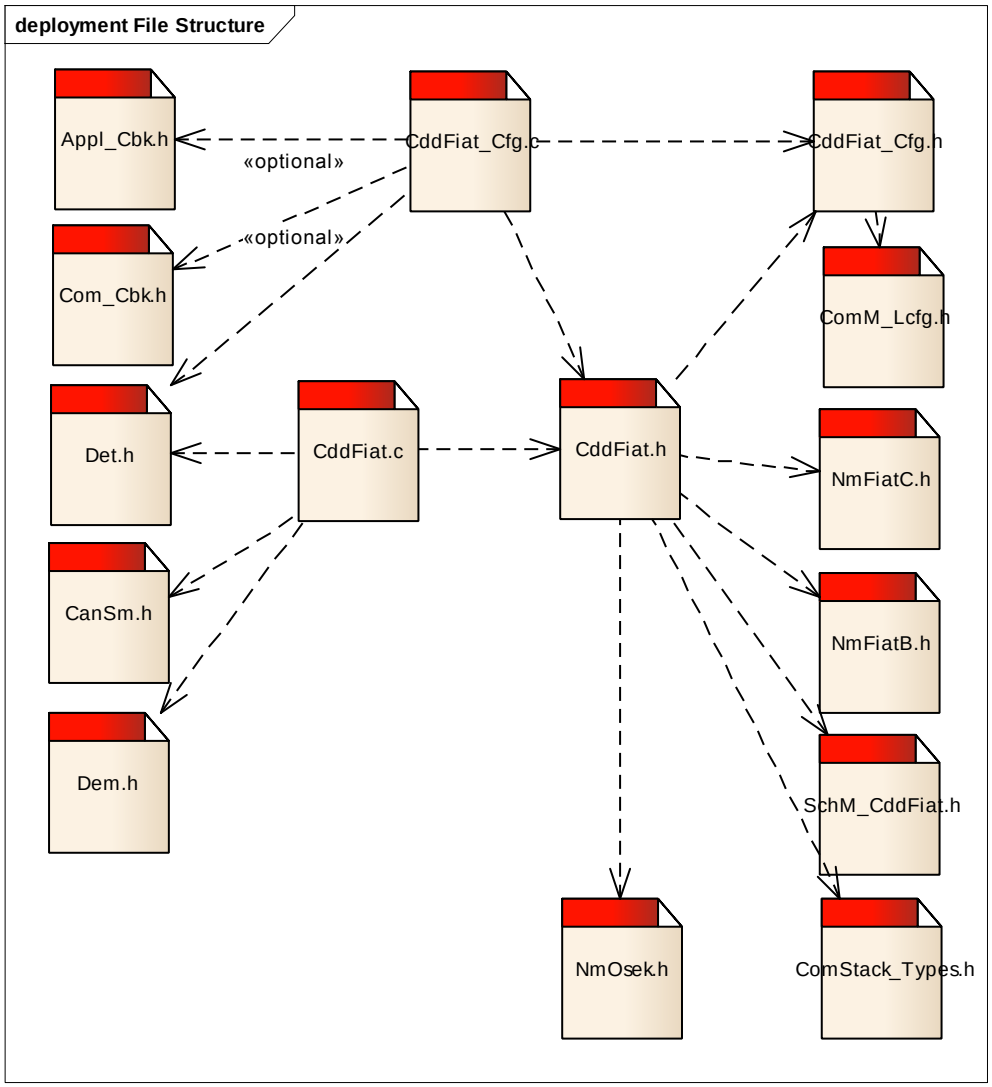

Figure 4-1 Include structure

4.3 Compiler Abstraction and Memory Mapping

The objects (e.g. variables, functions, constants) are declared by compiler independent definitions – the compiler abstraction definitions. Each compiler abstraction definition is assigned to a memory section.

The following table contains the memory section names and the compiler abstraction definitions of the CddFiat and illustrates their assignment among each other.

Compiler Abstraction Definitions / Memory Mapping Sections	CDDFIAT_CODE	CDDFIAT_APPL_CODE	CDDFIAT_VAR	CDDFIAT_VAR_NOINIT	CDDFIAT_CONST
CDDFIAT_START_SEC_VAR_NOINIT_8BIT CDDFIAT_STOP_SEC_VAR_NOINT_8BIT				■	
CDDFIAT_START_SEC_VAR_INIT_UNSPECIFIED CDDFIAT_STOP_SEC_VAR_INIT_UNSPECIFIED			■		
CDDFIAT_START_SEC_CODE CDDFIAT_START_SEC_CODE	■	■			
CDDFIAT_START_SEC_VAR_NOINIT_UNSPECIFIED CDDFIAT_STOP_SEC_VAR_NOINIT_UNSPECIFIED			■		
CDDFIAT_START_SEC_CONST_8BIT CDDFIAT_STOP_SEC_CONST_8BIT					■

Table 4-2 Compiler abstraction and memory mapping

4.4 Critical Sections

The CddFiat implements one critical section. It is used to protect all relevant read / modify / write accesses to internal variables. Within this critical section, interrupts have to be suspended.

Global Interrupt Operation	Function
Global Suspend Function	SchM_Enter_CddFiat(CDDFIAT_EXCLUSIVE_AREA_0)
Global Resume Function	SchM_Exit_CddFiat(CDDFIAT_EXCLUSIVE_AREA_0)

Table 4-3 Used functions for critical section handling

4.5 Initialization

The CddFiat has to be initialized by calling the API `void CddFiat_Init(void)`.The CddFiat component should be initialized before any involved component use any CddFiat API (see Chapter 4.6).

Caution
Initialize the CddFiat module before Dem, CanSM, Nm, NmFiatB, NmFiatC and Com were initialized. Otherwise the CddFiat may not be able to setup the content of the node status message correctly.

4.6 Dependencies on SW modules

4.6.1 Det

The Det component is used to report error during the development phase. This feature and therefore this dependency can be disabled via the component configuration.

4.6.2 Dem

The Dem is not used for reporting production errors as recommend within AUTOSAR components. Dem has to report node events to the CddFiat. Therefore the CddFiat implements the APIs `Appl_Dem_GenericFailStateChangeNotification` and `Appl_Dem_CurrentFailStateChangeNotification` as callout functions.

4.6.3 CanSM

The CanSM reports a bus off event to the CddFiat by using the API's `Appl_CanSM_BusOffBegin` and `Appl_CanSM_BusOffEnd`.

4.6.4 Com

For Class C and AUTOSAR Nm ECU's, the CddFiat implements the callout function `CddFiat_IPDU_Callout_<CHANNELNAME>` for each channel. The function will be called by Com before the PDU with the status message is sent. Within this function the CddFiat updates the corresponding status signals.

The API `CddFiat_BusOffCbkTAck_<CHANNELNAME>` is used to indicate the successful transmission of the BusOff signal.

4.6.5 SchM

Beside OSEK/AUTOSAR OS, the Basic Software Scheduler provides functions that the CddFiat calls at begin and end of critical sections.

5 API Description

For an interfaces overview please see Figure 2-2.

5.1 Services provided by CddFiat

5.1.1 CddFiat_Init

Prototype	
`void` **`CddFiat_Init`** `( void )`	
Parameter	
–	-
Return code	
–	-
Functional Description	
The function initializes the internal data structure.	
Particularities and Limitations	
■ This function is synchronous ■ This function is non-reentrant	
Expected Caller Context	
■ ECUM During Startup	

Table 5-1 CddFiat_Init

5.1.2 CddFiat_InitMemory

Prototype	
`void` **`CddFiat_InitMemory`** `(void)`	
Parameter	
–	-
Return code	
–	-
Functional Description	
Memory-Initialization of CddFiat. If RAM is not automatically initialized at start-up, this function must be called from start-up code and before CddFiat_Init is called to ensure that variables are initialized with a certain value if needed (e.g. initialization status with UNINIT value).	
Particularities and Limitations	
■ This function is synchronous ■ This function is non-reentrant	
Expected Caller Context	
■ The function should be called during startup	

Table 5-2 CddFiat_InitMemory

5.1.3 CddFiat_GetBusOffStatus

Prototype	
`Std_ReturnType` **`CddFiat_GetBusOffStatus`** `(NetworkHandleType nmChannelHandle, boolean *statePtr)`	
Parameter	
`nmChannelHandle`	Specifies the Com channels of which the bus off status should be gained
`statePtr`	Pointer to a boolean variable where the bus off status should be saved to.
Return code	
`E_OK`	No error has occurred.
`E_NOT_OK`	The bus off status could not be provided.
Functional Description	
The function provides the bus off status of a Com channel which is handled by the CddFiat. In case of an error the bus off status of the referenced variable will not be modified	
Particularities and Limitations	
■ The API is available for Class B ECU configuration ■ The API is available for Class C ECU configuration with wake-up algorithm enabled ■ This function is synchronous ■ This function is non-reentrant	
Expected Caller Context	
■ The function can be called in task and interrupt context	

Table 5-3 CddFiat_GetBusOffStatus

5.1.4 CddFiat_GetEOLStatus

Prototype	
`Std_ReturnType` **`CddFiat_GetEOLStatus`** `(boolean *statePtr)`	
Parameter	
`statePtr`	Pointer to a boolean variable where the end of line status should be saved to.
Return code	
`E_OK`	No error has occurred.
`E_NOT_OK`	The end of line status could not be provided.
Functional Description	
The function provides the EOL status of the ECU. In case of an error the EOL status of the referenced variable will not be modified	
Particularities and Limitations	
■ The API is only available for Class B ECU configuration ■ This function is synchronous ■ This function is non-reentrant	
Expected Caller Context	
■ The function can be called in task and interrupt context	

Table 5-4 CddFiat_GetEOLStatus

5.1.5 CddFiat_GetFailStatus

Prototype	
`Std_ReturnType` **`CddFiat_GetFailStatus`** `(boolean *genericFailStatePtr, boolean *currentFailStatePtr)`	
Parameter	
`genericFailStatePtr`	Pointer to a boolean variable where the generic Dem fail status should be saved to.
`currentFailStatePtr`	Pointer to a boolean variable where the current Dem fail status should be saved to.
Return code	
`E_OK`	No error has occurred.
`E_NOT_OK`	The Dem statuses could not be provided.
Functional Description	
The function provides the current and generic Dem fail status. In case of an error those statuses of the referenced variable will not be modified	
Particularities and Limitations	
■ The API is only available for Class B ECU configuration ■ This function is synchronous ■ This function is non-reentrant	
Expected Caller Context	
■ The function can be called in task and interrupt context	

Table 5-5 CddFiat_GetFailStatus

5.1.6 CddFiat_SetWakeReason

Prototype	
`Nm_ReturnType` **`CddFiat_SetWakeReason`** `(NetworkHandleType nmChannelHandle, uint8 WakeReason)`	
Parameter	
`nmChannelHandle`	Identification of the Nm-channel
`WakeReason`	Stores the current wake reason
Return code	
`NM_E_OK`	No error has occurred.
`NM_E_NOT_OK`	Setting the Wake Reason was not successful.
Functional Description	
This function can be called by the application to notify the current Wake Reason.	
Particularities and Limitations	
■ The API is only available for Class C ECU configuration with wake-up algorithm enabled ■ This function is synchronous ■ This function is non-reentrant	
Expected Caller Context	
■ The function can be called in task and interrupt context	

Table 5-6 CddFiat_SetWakeupReason

5.1.7 CddFiat_SetIgnitionStatus

Prototype	
`Nm_ReturnType` **`CddFiat_SetIgnitionStatus`** `(NetworkHandleType nmChannelHandle, uint8 IgnitionStatus)`	
Parameter	
`nmChannelHandle`	Identification of the Nm-channel
`WakeReason`	Stores the current Ignition Status
Return code	
`NM_E_OK`	No error has occurred.
`NM_E_NOT_OK`	Setting the Ignition Status was not successful.
Functional Description	
This function can be called by the application to set the current Ignition Status.	
Particularities and Limitations	
■ The API is only available for Class C ECU configuration with wake-up algorithm enabled ■ This function is synchronous ■ This function is non-reentrant	
Expected Caller Context	
■ The function can be called in task and interrupt context	

Table 5-7 CddFiat_SetIgnitionStatus

5.1.8 CddFiat_SetClassCSysConfig

Prototype
`Nm_ReturnType CddFiat_SetClassCSysConfig( NetworkHandleType nmChannelHandle, const uint8* const nmCSysConfPtr, const uint8 nmArraySize )`

Parameter	
`nmChannelHandle`	Channel ID of the physical channel
`nmCSysConfPtr`	Pointer to the Class C System Configuration Bit-Field
`nmArraySize`	Size of the given array

Return code	
`NM_E_OK`	No Error
`NM_E_NOK`	Failure due to an invalid parameter or component is not initialized.

Functional Description

This function should be called by application to set the current Class C System Configuration. The parameter nmCSysConfPtr should point to an array of size `nmArraySize`. The values are copied to an internal array. The least significant bit in byte 0 corresponds to the node with address 0. The most significant bit in byte (`nmArraySize` - 1) corresponds to the highest possible node address.

Note
This function is only available if Class B Master is active on at least one channel. Application must call this function on the Class B Master channels only.

Particularities and Limitations

- This function is synchronous
- This function is non-reentrant

Call context

- Function must be called from Task Level only.

Table 5-8 CddFiat_SetClassCSysConfig

5.1.9 CddFiat_StatusC_MessageIndication

Prototype	
`void CddFiat_StatusC_MessageIndication(NetworkHandleType nmChannelHandle)`	
Parameter	
`nmChannelHandle`	Channel ID of the physical channel
Return code	
–	-
Functional Description	
This function should be called by application to indicate a reception of a STATUC_C_<ECU> message of a non-wakeable slave. For configuration details please refer to Technical Reference of NmFiatC [7].	
Particularities and Limitations	
■ This function is synchronous ■ This function is reentrant	
Call context	
■ Function can be called from Task or Interrupt context	

Note
This function is only available if Class C Master is active on at least one channel.

Caution
This function must not be called on reception of a STATUS_C_<ECU> message from a wakeable Slave.

Table 5-9 CddFiat_StatusC_MessageIndication

5.2 Callback Functions

5.2.1 Appl_CanSM_BusOffBegin

Prototype	
`void` **`Appl_CanSM_BusOffBegin`** `(NetworkHandleType CanSM_NetworkHandle, CanSM_BorCounterType CanSM_BusOffNotificationCounter)`	
Parameter	
`CanSM_NetworkHandle`	Handle of the Network which reports the event
`CanSM_BusOffNotificationCounter`	Count of the occurred bus off events
Return code	
–	-
Functional Description	
The function notifies the occurrence of a bus off event on a given channel.	
Particularities and Limitations	
■ This function is synchronous ■ This function is non-reentrant	
Expected Caller Context	
■ The function should be called as a callback, for a occurred bus off event, by the CanSM component	

Table 5-10 Appl_CanSM_BusOffBegin

5.2.2 Appl_CanSM_BusOffEnd

Prototype	
`void` **`Appl_CanSM_BusOffEnd`** `(NetworkHandleType CanSM_NetworkHandle)`	
Parameter	
`CanSM_NetworkHandle`	Handle of the Network which reports the event
Return code	
–	-
Functional Description	
The function notifies the end of a bus off event.	
Particularities and Limitations	
■ This function is synchronous ■ This function is non-reentrant	
Expected Caller Context	
■ The function should be called as a callback, at the end of a bus off event, by the CanSM component	

Table 5-11 Appl_CanSM_BusOffEnd

5.2.3 Appl_Dem_GenericFailStateChangeNotification

Prototype	
`void` **`Appl_Dem_GenericFailStateChangeNotification`** `(boolean failState)`	
Parameter	
`failState`	Provides the information whether the Dem fail state is set or not
Return code	
–	-
Functional Description	
The function reports the actual generic Dem fail state.	
Particularities and Limitations	
■ This function is synchronous ■ This function is non-reentrant	
Expected Caller Context	
■ The function should be called as a callback by the Dem component	

Table 5-12 Appl_Dem_GenericFailStateChangeNotification

5.2.4 Appl_Dem_CurrentFailStateChangeNotification

Prototype	
`void` **`Appl_Dem_CurrentFailStateChangeNotification`** `(boolean failState)`	
Parameter	
`failState`	Provides the information whether the Dem fail state is set or not
Return code	
–	-
Functional Description	
The function reports the actual current Dem fail state.	
Particularities and Limitations	
■ This function is synchronous ■ This function is non-reentrant	
Expected Caller Context	
■ The function should be called as a callback by the Dem component	

Table 5-13 Appl_Dem_CurrentFailStateChangeNotification

5.3 Callout Functions

The declarations of the callout functions are provided by CddFiat module. It is the integrator's task to provide the corresponding function definitions. The definitions of the callouts can be adjusted to the system's needs. The CddFiat callout function declarations are described in the following tables:

5.3.1 Appl_CddFiat_GetEOLStatus

Prototype	
`boolean` **`Appl_CddFiat_GetEOLStatus`** `( void )`	
Parameter	
–	-
Return code	
`boolean`	The return values provide the EOL value of the ECU.
Functional Description	
The function provides the EOL status of the ECU. This API will only be called in case an AUTOSAR NM ECU is selected as used ECU variant.	
Particularities and Limitations	
■ This feature is only available for AUTOSAR Nm and Class B ECU's.	
Call context	
■ The component will call the API during initialization.	

Table 5-14 Appl_CddFiat_GetEOLStatus

5.3.2 Appl_CddFiat_SystemStatusIndication

Prototype	
`void` **`Appl_CddFiat_SystemStatusIndication`** `(NetworkHandleType nmChannelHandle, NmFiatB_SystemConfigType currentSysConfig, uint8 arraySize)`	
Parameter	
`nmChannelHandle`	Identification of the Nm-channel
`currentSysConfig`	Pointer to the current System Configuration
`arraySize`	Size of the given array
Return code	
–	-
Functional Description	
This callback is called on Class B Master channels only. It indicates the current monitored system configuration to application. See 3.2 System Status Indication.	
Particularities and Limitations	
■ This function is synchronous ■ This function is non-reentrant	
Expected Caller Context	
■ Function is called on task level only.	

Table 5-15 Appl_CddFiat_SystemStatusIndication

5.3.3 Appl_CddFiat_GetTrcvErrorPin

Prototype	
`boolean` **`Appl_CddFiat_GetTrcvErrorPin`** `(NetworkHandleType nmChannelHandle)`	
Parameter	
`nmChannelHandle`	Identification of the Nm-channel
Return code	
`boolean`	TRUE if the transceiver indicated an error during message reception. FALSE if there was no error during message reception.
Functional Description	
The function is used for “CAN bus wire failure detection” as described in [4]. It shall return TRUE if the transceiver indicated an error during message reception.	
Particularities and Limitations	
■ This function is only available for Class B Master. ■ This function is synchronous ■ This function is non-reentrant	
Expected Caller Context	
■ This function is called within NmFiatB_RxIndication (Interrupt context)	

Table 5-16 Appl_CddFiat_GetTrcvErrorPin

5.4 Services used by CddFiat

The CddFiat uses the following components’ services. For details about prototype and functionality, refer to the documentation of the providing component.

Component	API
Det	`Det_ReportError` (optionally)

Table 5-17 Services used by CddFiat

5.5 Configurable Interfaces

5.5.1 Callback Functions

The following callbacks are generated by the module CddFiat. For each configured channel which needs this callback, an own function will be generated with the name of the channel as postfix.

5.5.1.1 CddFiat_BusOffCbkTAck_<CHANNELNAME>

Prototype	
`void` **`CddFiat_BusOffCbkTAck_<CHANNELNAME>`** `( void )`	
Parameter	
–	-
Return code	
–	-
Functional Description	
The function notifies the correct transmission of the bus off signal for the configured communication channel. This function is generated by GENy and needs to be configured as described in 7.1.3.2.	
Particularities and Limitations	
■ The API is only available for Class C and AUTOSAR Nm ECU's ■ This function is synchronous ■ This function is non-reentrant	
Expected Caller Context	
■ This function should be set up as signal transmission confirmation callback within the Com module for the referenced bus off signal	

Table 5-18 CddFiat_BusOffCbkTAck_<CHANNELNAME>

5.5.2 Callout Functions

The following callouts are generated by the module CddFiat. For each configured channel which needs this callout, an own function will be generated with the name of the channel as postfix.

5.5.2.1 CddFiat_IPDU_Callout_<CHANNELNAME>

Prototype	
`void` **`CddFiat_IPDU_Callout_<CHANNELNAME>`** `( PduIdType ID, const PduInfoType * PduInfoPtr)`	
Parameter	
`ID`	Id of the IPDU message to be transmitted
`PduInfoPtr`	Pointer to the IPDU information structure owning the message buffer reference
Return code	
–	-
Functional Description	
This function notifies the transmission of the signals which will be updated by the CddFiat for this communication channel. This function is generated by GENy and needs to be configured as described in 7.1.3.1.	
Particularities and Limitations	
■ The API is only available for Class C and AUTOSAR Nm ECU's ■ This function is synchronous ■ This function is non-reentrant	
Expected Caller Context	
■ The function should be called as IPDU call back by the Com for the referenced IPDU message	

Table 5-19 CddFiat_IPDU_Callout_<CHANNELNAME>

6 Signal Description

This chapter describes the signals which will be updated by CddFiat. The signals are provided by the FGA message map. Please note: the name of the message and signals may vary depending on the FGA dbc file.

6.1 Signal Description AUTOSAR ECU

Signal Name	Signal Location	Description
D_ES	PSC_INFO_<ECU>, Bit position 60	Signal provides the information if a bus off has occurred
CurrentFailSts	PSC_INFO_<ECU>, Bit position 62	Signal provides the current fail state information of the component Dem
GenericFailSts	PSC_INFO_<ECU>, Bit position 61	Signal provides the generic fail state information of the component Dem
EOL	PSC_INFO_<ECU>, Bit position 63	Provides the information if a EOL programming was done by Fiat or not

Table 6-1 Node status message signal description for AUTOSAR ECUs

6.2 Signal Description Class C ECU

The signal positions of the node status message are in line with FGA norm TFO07321.

Signal Name	Signal Location	Description
D_ES	STATUS_C_xxx message, Bit position 60-59	Signal provides the information if a bus off has occurred
EOL	STATUS_C_xxx message, Bit position 58	Provides the information if a EOL programming was done by Fiat or not
CurrentFailSts	STATUS_C_xxx message, Bit position 57	Signal provides the current fail state information of the component Dem
GenericFailSts	STATUS_C_xxx message, Bit position 56	Signal provides the generic fail state information of the component Dem

Table 6-2 Node status message signal description for Carry-Back Class C ECUs

6.3 Signal Description Class B ECU

The Nm message of Class B Nm ECUs is composed by the MICROSAR Class B Nm module provided by Vector Informatik. The CddFiat module provides the relevant information to the Class B Nm module.

6.4 Signals not handled by the CddFiat component

The CddFiat only handles signals defined in the section above. All remaining signals of the node status message (such as the StayActiveSource signal) must be handled by the application by using standard signal access mechanisms of Com or RTE.

7 Configuration

7.1 Configuration with GENy

The CddFiat module is configured with the help of the configuration tool GENy.

7.1.1 Activation of the module CddFiat in GENy

To activate the module CddFiat at least one communication channel must be selected on the Component Selection page as shown in Figure 7-1.

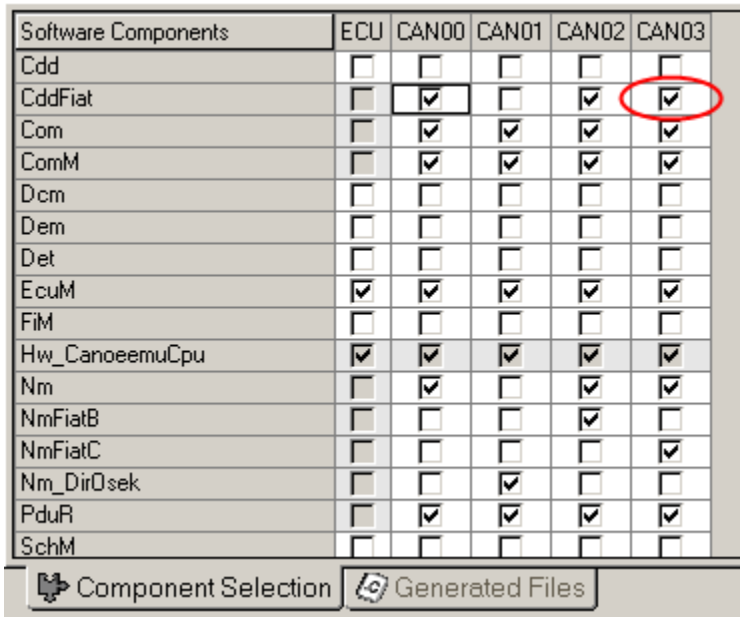

Figure 7-1 Activation of the module CddFiat

7.1.2 General Configuration

This chapter describes the general configuration for the module CddFiat. Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
Common				
Configuration Variant	-	-	**Pre-Compile**	Specify the configuration variant
Dev Error Detect	pre-compile	boolean	ON **OFF**	If 'Development Error Detection' is ON, development errors are reported to the Development Error Tracer (Det).
User Configuration	pre-compile	String		A CddFiat configuration file is generated by GENy

This chapter describes the general configuration for the module CddFiat. Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
File Path				(CddFiat_cfg.h). If you want to overwrite settings in the generated file, you can specify a path to a user defined configuration file. The user defined configuration file will be included at the end of the generated file. Therefore definitions in the user defined configuration file can overwrite definitions in the generated configuration file.

Table 7-1 General Configuration

7.1.3 Channel Configuration

This chapter describes the channel configuration of the CddFiat. For Class B Nm channels this fields can be left empty because this is not necessary for Class B.

For each CAN communication channel that requires FGA node status information (e.g. EOL Flag) to be transmitted some configurations have to be made to allow CddFiat to work.

7.1.3.1 Configuration of the IPdu Callout

For each configured Class C or Autosar Nm channel an IPdu Callout must be configured.

In a drop-down list Tx IPdus are listed which are configured for the selected channel. You have to choose the IPdu which transmits the EOL, current generic fail states and BusOff status signals. Please refer to the FGA message map to locate this message for your ECU.

In Multiple Ecu configurations the IPdu has to be chosen for each identity.

Figure 7-2 shows a screenshot were the relevant set up was done.

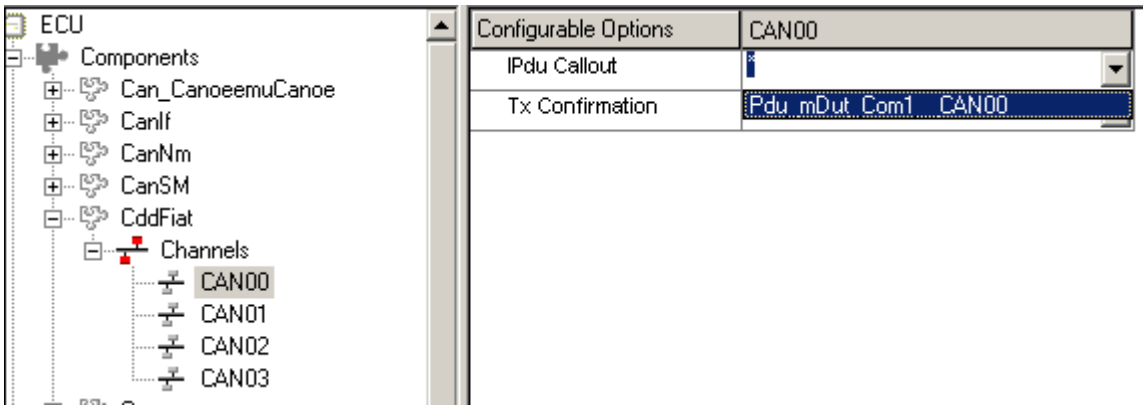

Figure 7-2 Configuration of the IPdu Callout

The function will be generated to the file CddFiat_Cfg.c and the prototype will be generated to Com_Cbk.h.

7.1.3.2 Configuration of the Tx Transmit Callback

The CddFiat requires a Tx confirmation function which indicates successful transmission of the node status message. In AUTOSAR the Tx confirmation needs to be configured on signal level.

Choose one single signal out of the configured IPdu. After successful transmission of the node status message the Com calls a Tx confirmation function which is generated by the module CddFiat to reset the BusOff information after it was send one time.

In Multiple Ecu configurations the Tx signal has to be chosen for each identity.

Figure 7-3 shows a screenshot were the relevant set up was done.

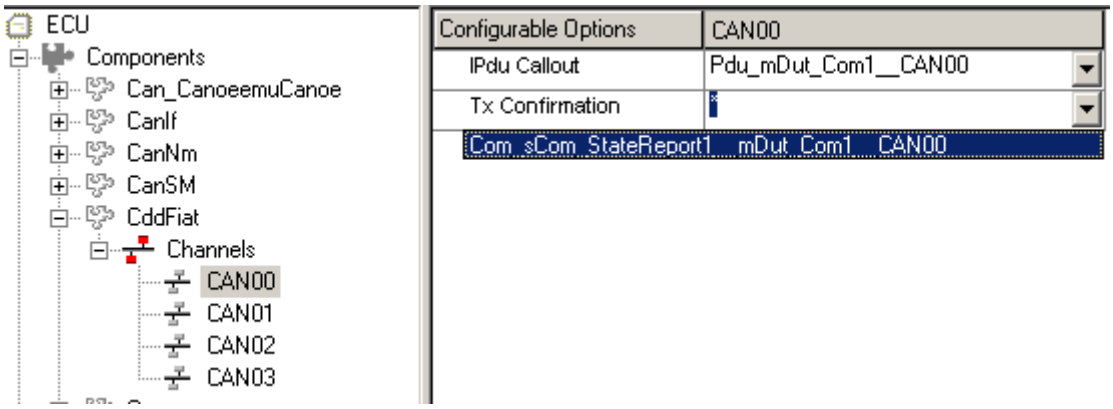

Figure 7-3 Configuration of the Tx Confirmation

The function will be generated to the file CddFiat_Cfg.c and the prototype will be generated to Rte_Cbk.h or Appl_Cbk.h.

8 Abbreviations

8.1 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
CanSM	CAN State Manager
CDD	Complex Device Driver
Dem	Diagnostic Event Manager
Det	Development Error Tracer
ECU	Electronic Control Unit
EOL	End Of Line
SWS	Software Specification

Table 8-1 Abbreviations

9 Contact

Visit our website for more information on

- News
- Products
- Demo software
- Support
- Training data
- Addresses

www.vector.com