MICROSAR CAN State Manager

Technical Reference

Version 1.16

Authors	Mark A. Fingerle
Status	Released

1 Document Information

1.1 History

Author	Date	Version	Remarks
Mark A. Fingerle	2008-02-04	1.0	ASR 3.0 beta release
Mark A. Fingerle	2008-02-27	1.1	Switch to new template
Mark A. Fingerle	2008-03-31	1.2	ESCAN00025504 Rename Technical Reference to MSR Short Name ESCAN00025710 Adapt initialization. ESCAN00025711 Adapt Chapter Configuration GENy to ASR names
Mark A. Fingerle	2008-05-06	1.3	ESCAN00025892 Simplify the set controller mode algorithm of the CanSM ESCAN00026001 Extend error handling to the case a transition fails and the mode request changes before recovering of the transition.
Mark A. Fingerle	2008-07-18	1.4	Allowed timer values IPDU via drop down list critical section kinds
Mark A. Fingerle	2008-10-08	1.5	New feature/API ECU passive mode CAN bus specific bus-off configuration parameter
Mark A. Fingerle	2008-10-28	1.6	State machine start/entry point Additional explanation of passive mode Additional explanation of error counter Figure 4-1 updated GENy screen shots updated
Mark A. Fingerle	2009-02-18	1.7	add API CanSM_PreventBusSleepAtStartUp advance chapter 4.2 Initialization
Mark A. Fingerle	2009-03-23	1.8	Adapt configuration of the transceiver handling
Mark A. Fingerle	2009-05-13	1.9	Correct Figure 6-1 CanSM interactions with other BSW 4.3.1 “Transition Check” in case set transceiver mode fail Add Error code `CANSM_E_SETTRANSCEIVERMODE` in Table 4-6 Add application bus-off notification functions 4.3.2, 6.5.2, 6.5.2
Mark A. Fingerle	2009-10-13	1.10	ESCAN00037731 Ambiguous description of critical sections in chapter 5.5 GENy feature **Disable Pdu Group** and **BusOff Notification**

			7.1.5
Mark A. Fingerle	2010-01-13	1.10.01	Correct chapter 7.1.3, EcuM ➔ SchM
Mark A. Fingerle	2010-03-03	1.11	Update 5.5 critical sections
Mark A. Fingerle	2010-04-23	1.12	ESCAN00040930 Add API EcuM_GeneratorCompatibilityError Table 6-11, Figure 6-1 ESCAN00037126 Better the Onscreen Help of the "Bor Counter L2 Err Ch" Table 7-4 ESCAN00041444 re-initialization of signals when switching from NO to FULL communication in chapter 7.1.5
Mark A. Fingerle	2010-08-13	1.13	ESCAN00043552 Add support for XCP shutdown chapter 4.4
Mark A. Fingerle	2010-10-03	1.14	ESCAN00045704 Disable DeadlineMonitoring in state CANSM_SILENT_COMMUNICATION. Add DM description to the states in chapter 4.3.1. ESCAN00045482 Wrong configuration class for AUTOSAR parameter: CanSM Post Build Config Start Address Table 7-1
Mark A. Fingerle	2011-01-23	1.15	R11: During the BusOff recovery the Nm may trigger the shutdown NmTimeOut 4.3.2 BusOffEnd missing if NoCom is triggered 4.3.2 Partial Networks 8.1.10 BswM ComMode Indication 8.1.11 Multiple identity 8.1.12 Usage `CANSM_EXCLUSIVE_AREA_4`
Mark A. Fingerle	2011-04-23	1.16	R12: ESCAN00049972 Add MSR Dem errors to deviation chapter 8.1.13 ESCAN00047985 No CAN communication possible, caused by bus-off in SILENT communication 8.1.14 ESCAN00050250 Extend Bus Off recovery handling, additional ModeIndications during BusOff recovery 4.3.2

Table 1-1 History of the document

1.2 Reference Documents

No.	Title	Version
[1]	AUTOSAR Specification of CAN State Manager	1.0.0
[2]	AUTOSAR Specification of Development Error Tracer	2.2.0
[3]	AUTOSAR Specification of Diagnostics Event Manager	2.2.1
[4]	AUTOSAR List of Basic Software Modules	1.2.0
[5]	AUTOSAR Specification of CAN Interface	2.1.0

[6]	AUTOSAR Specification of Communication Manager	2.0.0
[7]	AN-ISC-8-1093	1.0.0
[8]	AN-ISC-8-1118 MICROSAR BSW Compatibility Check	1.0.0

Table 1-2 Reference documents

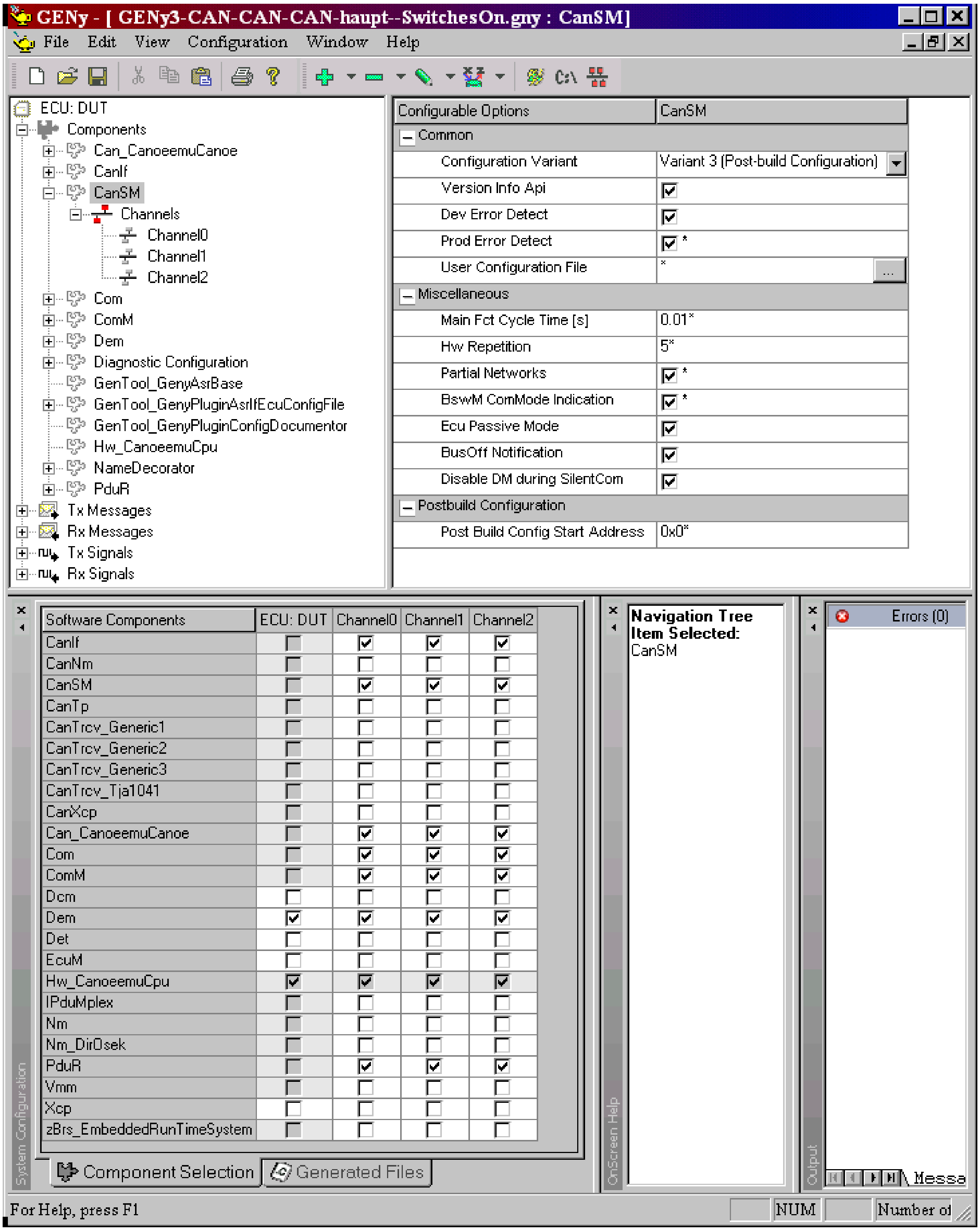
1.3 Scope of the Document

This technical reference describes the general use of the CAN State Manager basis software. All aspects which are CAN controller specific are described in a separate document [5], which is also part of the delivery.

⚠	**Please note** We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector´s release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Contents

Illustrations



ECU: DUT
- Components
  - Can_CanoeemuCanoe
  - CanIf
  - CanSM
    - Channels
      - Channel0
      - Channel1
      - Channel2
  - Com
  - ComM
  - Dem
  - Diagnostic Configuration
  - GenTool_GenyAsrBase
  - GenTool_GenyPluginAsrIfEcuConfigFile
  - GenTool_GenyPluginConfigDocumentor
  - Hw_CanoeemuCpu
  - NameDecorator
  - PduR
- Tx Messages
- Rx Messages
- Tx Signals
- Rx Signals

Configurable Options	CanSM
Common	
Configuration Variant	Variant 3 (Post-build Configuration)
Version Info Api	✓
Dev Error Detect	✓
Prod Error Detect	✓ *
User Configuration File	*
Miscellaneous	
Main Fct Cycle Time [s]	0.01*
Hw Repetition	5*
Partial Networks	✓ *
BswM ComMode Indication	✓ *
Ecu Passive Mode	✓
BusOff Notification	✓
Disable DM during SilentCom	✓
Postbuild Configuration	
Post Build Config Start Address	0x0*

Software Components	ECU: DUT	Channel0	Channel1	Channel2
CanIf	☐	✓	✓	✓
CanNm	☐	☐	☐	☐
CanSM	☐	✓	✓	✓
CanTp	☐	☐	☐	☐
CanTrcv_Generic1	☐	☐	☐	☐
CanTrcv_Generic2	☐	☐	☐	☐
CanTrcv_Generic3	☐	☐	☐	☐
CanTrcv_Tja1041	☐	☐	☐	☐
CanXcp	☐	☐	☐	☐
Can_CanoeemuCanoe	☐	✓	✓	✓
Com	☐	✓	✓	✓
ComM	☐	✓	✓	✓
Dcm	☐	☐	☐	☐
Dem	✓	✓	✓	✓
Det	☐	☐	☐	☐
EcuM	☐	☐	☐	☐
Hw_CanoeemuCpu	✓	✓	✓	✓
IPduMplex	☐	☐	☐	☐
Nm	☐	☐	☐	☐
Nm_DirOsek	☐	☐	☐	☐
PduR	☐	✓	✓	✓
Vmm	☐	☐	☐	☐
Xcp	☐	☐	☐	☐
zBrs_EmbeddedRunTimeSystem	☐	☐	☐	☐

Navigation Tree Item Selected:
CanSM

Tables

2 Component History

New Features
ASR 3 release of the CAN State Manager implementation
API ECU passive mode
Prevent bus sleep at startup
Application bus-off notification

Table 2-1 Component history

3 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module CanSM as specified in [1].

Supported AUTOSAR Release*:	3	
Supported Configuration Variants:	pre-compile, link-time, post-build	
Vendor ID:	CANSM_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	CANSM_MODULE_ID	`0x8C` (according to ref. [4])

* For the precise AUTOSAR Release 3.x please see the release specific documentation.

The CAN State Manager (CanSM) realizes a software layer between the Communication Manager (ComM) and the CAN Interface (CanIf). The CanSM handles the startup and shutdown of the communication of a CAN network. The CAN State Manager maps the CAN State Manager states to the states of the ComM and causes the necessary actions to change the CAN State Manager state to those requested by the ComM. The main function of the CAN State Manager is called cyclically by the Schedule Manager (SchM).

3.1 Architecture Overview

The following figure shows where the CanSM is located in the AUTOSAR architecture.

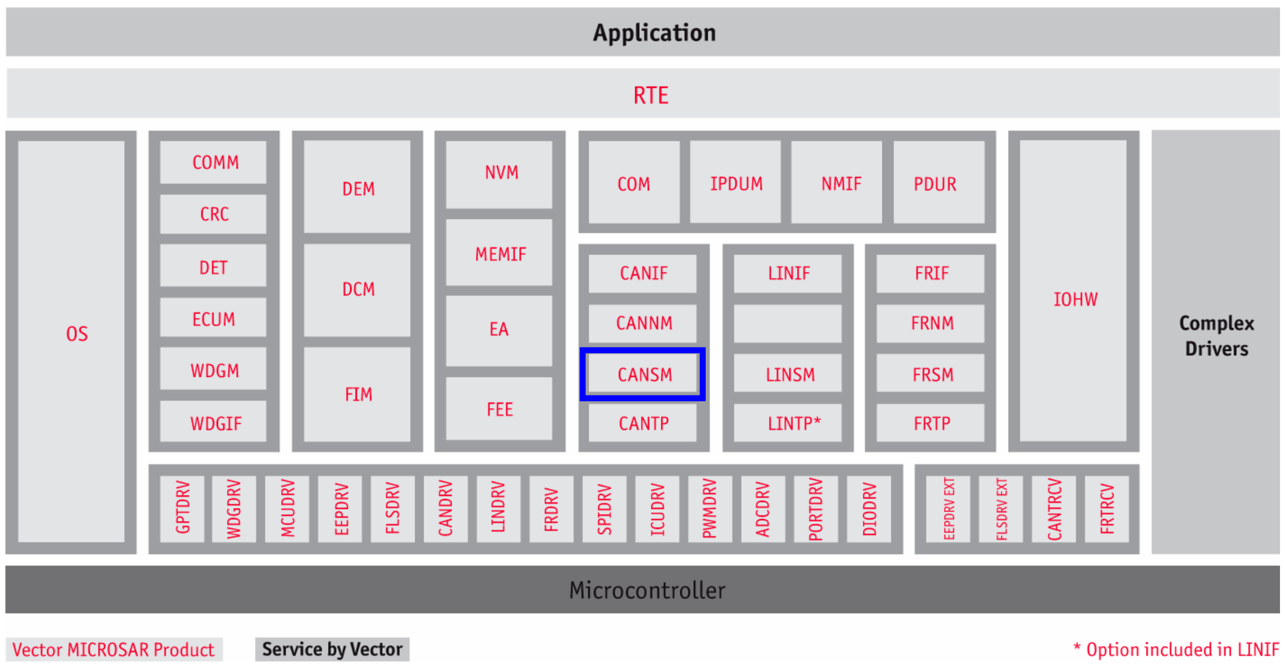


Figure 3-1 AUTOSAR architecture

The next figure shows the interfaces to adjacent modules of the CanSM. These interfaces are described in chapter 6.

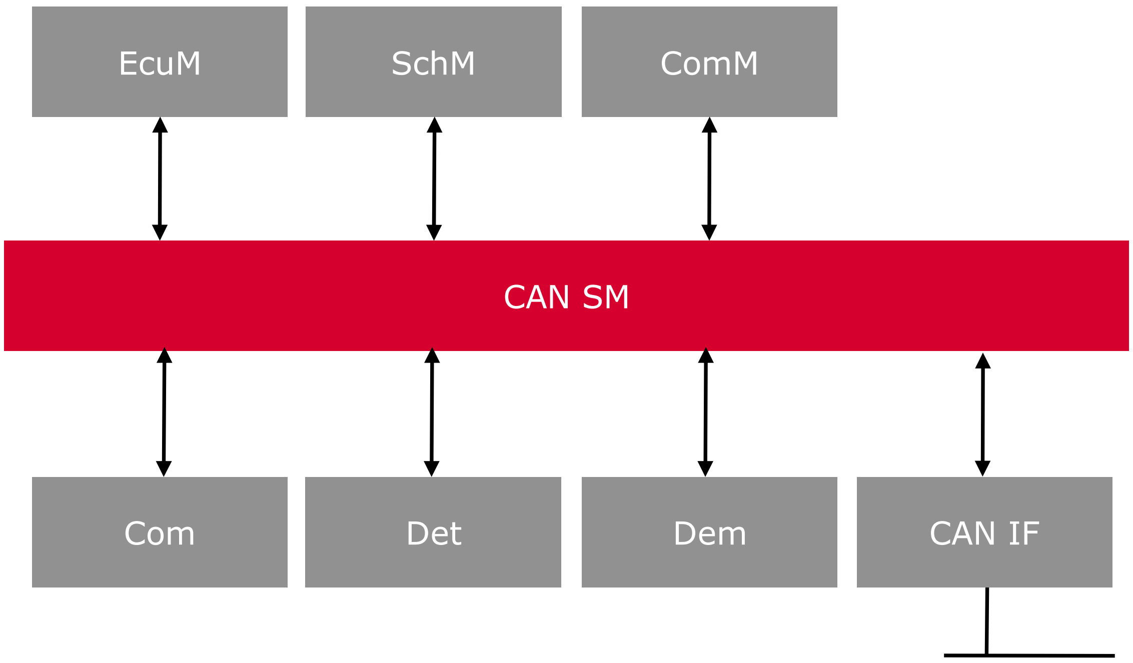


Figure 3-2 Interfaces to adjacent modules of the CanSM

SWC do not access the services of the BSW modules directly. They use the service ports provided by the BSW modules via the RTE. The services provided by the CanSM are listed in chapter 6.3 and are defined in [1].

4 Functional Description

4.1 Features

The features listed in this chapter cover the complete functionality specified in [1].

The "supported" and "not supported" features are presented in the following two tables. For further information of not supported features also see chapter 8.2.

The following features described in [1] are supported:

Feature
Translation of network communication mode requests
Output of current network communication modes (Polling and Callback)
Control of peripherals (CAN Transceivers, CAN Controllers)
Control of PDU mode
Control of PDU groups
Handle the network mode via a separate state machine per network
Bus error management: Bus-off recovery via a separate state machine per network
Error classification detection notification
Enable and disable development and production error detection
Handle several CAN networks

Table 4-1 Supported SWS features

The following features described in [1] are not supported:

Feature
The CanSM networks and CanSM controllers are not changeable via Post-build configuration.
The CanSM supports several controllers per network.
The CanSM supports Post-build process with set of multiple parameters

Table 4-2 Not supported SWS features

4.2 Initialization

Some embedded targets don’t (re)initialize variables correctly. Therefore some variables have to be initialized explicitly if they need a specific value before the initialization function `CanSM_Init` is called. This is done by the function `CanSM_InitMemory(void)`. The function initializes the CanSM variables and sets the state to un-init. The function has to be called before the initialization function.

After that the initialization `CanSM_Init` has to be triggered and the CAN State Manager will set the internal used variables to their start values to ensure a deterministic behavior of the state machines. The state machines become initialized, too. If selectable post build configuration is active a valid configuration has to be passed. In other cases the function has to be called without a parameter and the init function will map the configuration data to a pointer.

	Info The CanSM initializes the CAN channel into the state NO COMMUNICATION. This means, the CAN modules (CanIf, CanDrv and CanTrcv) are set into the corresponding state for NO COMMUNCIATION (bus sleep). During this transition, detected wake up reasons, inside the CAN modules, are cleared. This leads to the behavior that wake up events, which are triggered by the CAN bus, can not be detected and/or validated during the initialization phase. If the detection/validation of the wake up information is necessary for the ECU then the CanSM API CanSM_PreventBusSleepAtStartUp() can be used to prevent the bus sleep mode at start up for the above listed CAN modules. For further information, refer to [7].

4.3 State Machines

The CanSM contains two different state machines per network. Each CAN network gets its own network mode state machine and its own bus-off state machine.

4.3.1 Network Mode State Machine

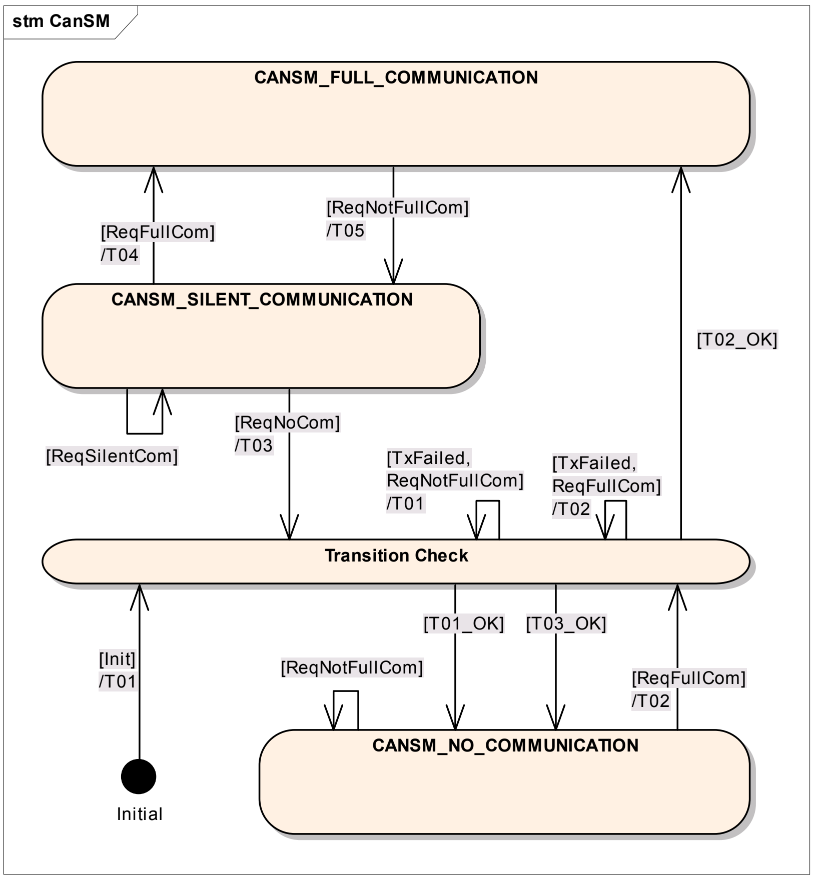


Figure 4-1 Network mode state machine

The network mode state machine handles the activation and the deactivation of the CAN communication.

Initial

The CAN state manager functionality can not be used before the API function `CanSM_Init` has been called. If the CanSM_Init function is executed successfully the CanSM performs the transition T01 and sets the state to `CANSM_NO_COMMUNNICATION`.

Transition Check

If the transition T01, T02 or T03 succeeds the state “Transition Check” is skipped and the CanSM switches to the following state, in the same cycle.

The CanSM evaluates if the desired controller mode is reached in T01, T02 and T03 and if the desired transceiver mode is reached in T02 only. If the CAN state manager detects a failure the CanSM enters the virtual state “Transition Check”, increases the error counter `Hw Repetition` (see Table 7-1) and the further actions of the transition are skipped. Afterwards the CanSM cyclically tries to reach the desired state until the transition passes. Therefore the CanSM triggers the transition T01 or T02 (see Figure 4-1), depending on the requested communication mode. Each sequently failed transition is counted and if the specified number exceeds the CanSM triggers the DEM failed message `CANSM_E_MODE_CHANGE_NETWORK_x`. The counter is reset if a transition succeeds.

If the function CanSM_GetCurrentComMode is called and the CanSM is in state "Transition Check" the prior state is returned as the current communication mode, because the state Transition Check" is unknown by the other modules.

CANSM_NO_COMMUNICATION

In this state there is no communication on the CAN channel. When full communication is requested the CAN state manager starts the transceiver and PDU groups. Additionally the BOR SM switches to the Meta state "Recovery enabled" (see 4.3.2). When the wake up process is not finished successfully (e.g. a used BSW function reports an error) the CAN state manager stays in the state and retries the transition next cycle if full communication is still requested. In case of a successful transition the CAN state manager notifies the communication manager about the new communication state and enables the deadline monitoring, if the feature is activated in the module Com.

CANSM_FULL_COMMUNICATION

The CanSM stays in this state as long as full communication is requested. Otherwise the CanSM switches to silent mode and stops the PDU groups. In case of a successful transition the CAN state manager notifies the communication manager about the new communication state. The BOR switches to the Meta state "Recovery disabled" and switches the deadline monitoring, if the feature is activated in the module Com. By default the deadline monitoring is activated. Via a configuration switch the CanSM can also be disposed to deactivate the monitoring in state CANSM_SILENT_COMMUNICATION to avoid Rx timeouts.

CANSM_SILENT_COMMUNICATION

The state represents the prepare bus sleep phase of the network. The node is still able to receive CAN messages but doesn't transmit them. According to the requested communication mode the CanSM switches to `CANSM_NO_COMMUNNICATION`, back to `CANSM_FULL_COMMUNICATION` or stays in the state. If the deadline monitoring is activated in the module Com, the CanSM ensures that the deadline monitoring of the Com is activated if `CANSM_FULL_COMMUNICATION` reached and deactivated if `CANSM_NO_COMMUNICATION` if reached.

4.3.2 Bus-off Recovery State Machine

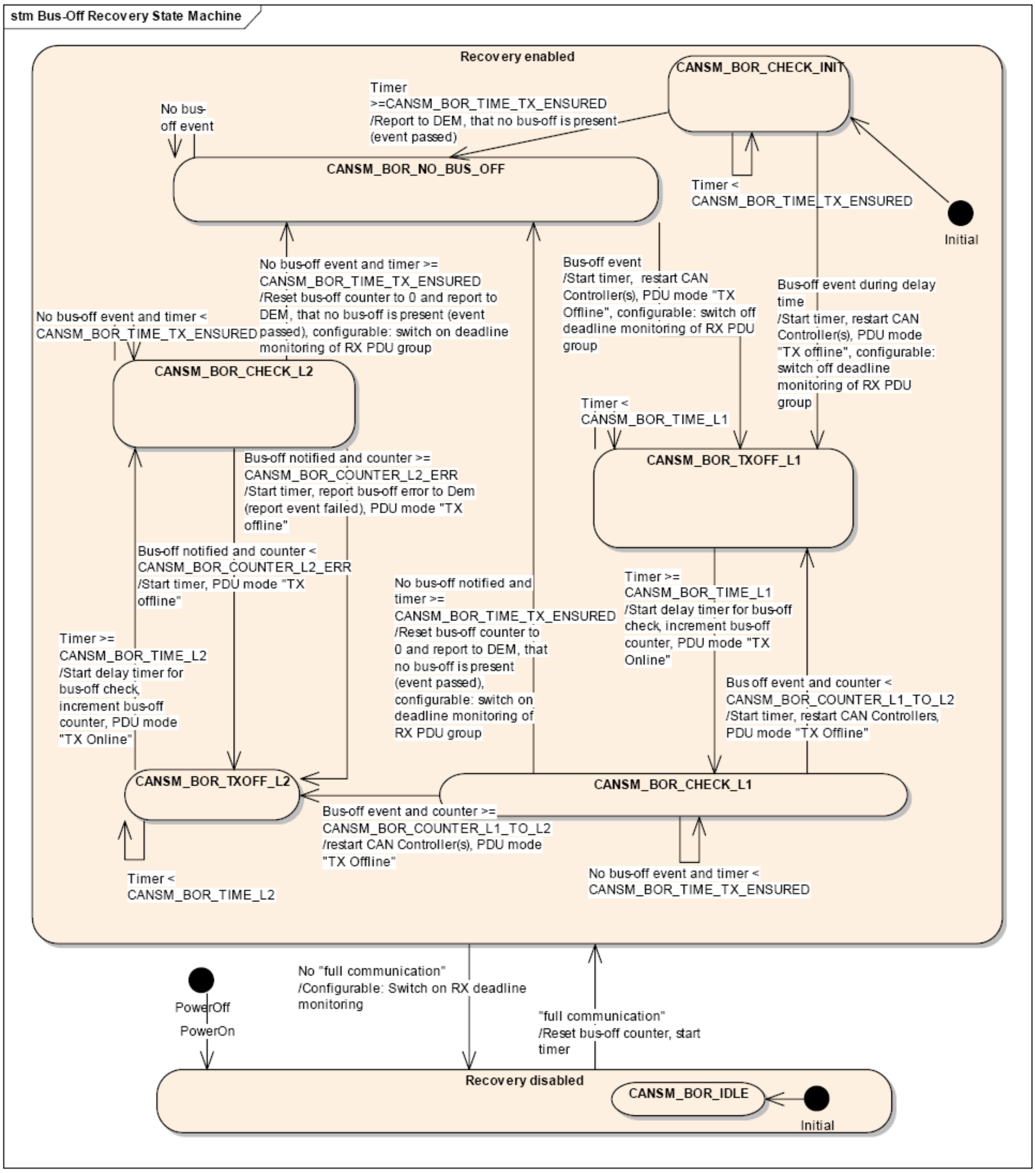

stm Bus-Off Recovery State Machine
Recovery enabled
CANSM_BOR_CHECK_INIT
Timer >=CANSM_BOR_TIME_TX_ENSURED /Report to DEM, that no bus-off is present (event passed)
No bus-off event
CANSM_BOR_NO_BUS_OFF
Timer < CANSM_BOR_TIME_TX_ENSURED
Initial
No bus-off event and timer >= CANSM_BOR_TIME_TX_ENSURED /Reset bus-off counter to 0 and report to DEM, that no bus-off is present (event passed), configurable: switch on deadline monitoring of RX PDU group
No bus-off event and timer < CANSM_BOR_TIME_TX_ENSURED
Bus-off event /Start timer, restart CAN Controller(s), PDU mode "TX Offline", configurable: switch off deadline monitoring of RX PDU group
Bus-off event during delay time /Start timer, restart CAN Controller(s), PDU mode "TX offline", configurable: switch off deadline monitoring of RX PDU group
CANSM_BOR_CHECK_L2
Timer < CANSM_BOR_TIME_L1
CANSM_BOR_TXOFF_L1
Bus-off notified and counter >= CANSM_BOR_COUNTER_L2_ERR /Start timer, report bus-off error to Dem (report event failed), PDU mode "TX offline"
Bus-off notified and counter < CANSM_BOR_COUNTER_L2_ERR /Start timer, PDU mode "TX offline"
No bus-off notified and timer >= CANSM_BOR_TIME_TX_ENSURED /Reset bus-off counter to 0 and report to DEM, that no bus-off is present (event passed), configurable: switch on deadline monitoring of RX PDU group
Timer >= CANSM_BOR_TIME_L1 /Start delay timer for bus-off check, increment bus-off counter, PDU mode "TX Online"
Timer >= CANSM_BOR_TIME_L2 /Start delay timer for bus-off check, increment bus-off counter, PDU mode "TX Online"
Bus off event and counter < CANSM_BOR_COUNTER_L1_TO_L2 /Start timer, restart CAN Controllers, PDU mode "TX Offline"
CANSM_BOR_TXOFF_L2
CANSM_BOR_CHECK_L1
Bus-off event and counter >= CANSM_BOR_COUNTER_L1_TO_L2 /restart CAN Controller(s), PDU mode "TX Offline"
Timer < CANSM_BOR_TIME_L2
No bus-off event and timer < CANSM_BOR_TIME_TX_ENSURED
No "full communication" /Configurable: Switch on RX deadline monitoring
PowerOff
PowerOn
"full communication" /Reset bus-off counter, start timer
Recovery disabled
CANSM_BOR_IDLE
Initial

Figure 4-2 Bus-off recovery state machine

The BOR handles the restart of the CAN communication after a bus-off. The BOR SM becomes activated, meta state “Recovery enabled”, if the network state machine reaches full communication and the BOR SM is deactivated if the full communication state is left, state `CANSM_BOR_IDLE`.

Transition to Meta state “Recovery disabled” see state CANSM_FULL_COMMUNICATION in chapter 4.3.1.

CANSM_BOR_IDLE

The Network Mode State Machine is not in `CANSM_FULL_COMMUNICATION`. BOR timer and counter are deactivated.

CANSM_BOR_CHECK_INIT

The CanSM waits for a bus-off until timer `TxEnsured` exceeds. If during this period no bus-off is reported, the BOR SM switches to `CANSM_BOR_NO_BUS_OFF`. Otherwise the controller is restarted, the deadline monitoring becomes deactivated and the CanSM switches to `CANSM_BOR_TXOFF_L1`.

CANSM_BOR_NO_BUS_OFF

The Network Mode State Machine is in `CANSM_FULL_COMMUNICATION` and no error is detected. The CAN bus has the full functionality, is able to receive and send messages. Each time the state is entered a DEM `DEM_EVENT_STATUS_PASSED` is send.

If a bus-off is reported the controller becomes restarted, the deadline monitoring becomes deactivated and the CanSM switches to `CANSM_BOR_TXOFF_L1`.

CANSM_BOR_TXOFF_L1, CANSM_BOR_TXOFF_L2

In these states the node transmits no messages on the bus.

If the `Level 2 recovery time` has expired the Can SM switches to `CANSM_BOR_CHECK_Lx` and activates the transmission of messages again.

On entering one of these states the CanSM will call the function `ComM_BusSM_ModeIndication`, report SILENT_COMMUNICATION mode to the ComM module and FULL_COMMUNICATION mode again on exit. During the CanSM is in state TXOFF_L1 (or TXOFF_L2) the function `CanSM_GetCurrentComMode` reports SILENT_COMMUNICATION mode.

CANSM_BOR_CHECK_L1, CANSM_BOR_CHECK_L2

The CanSM waits for a bus-off until timer `TxEnsured` exceeds. If no bus-off is reported during this period the BOR SM switches to `CANSM_BOR_NO_BUS_OFF`. Otherwise the controller becomes restarted, the transmission of messages is deactivated and the error counter is increased.

The following state depends on the error counter. If the value of the error counter is lower than the threshold `Counter L1 to L2` the CanSM switches to `CANSM_BOR_TXOFF_L1`. Otherwise the next state is `CANSM_BOR_TXOFF_L2`.

If the value of the error counter reaches the threshold `L2 error counter` the CanSM additionally triggers DEM `DEM_EVENT_STATUS_FAILED`. CanSM still continues with the BOR mechanism and each bus-off causes a DEM failed until the BOR is successful or the SWC reacts and sets the CAN to no communication or silent communication.

Bus-off notification functions Appl_CanSM_BusOffBegin and Appl_CanSM_BusOffEnd

The feature gives the application the possibility to react on an active bus-off. If the feature is activated the CanSM triggers the `Appl_CanSM_BusOffBegin` immediately and each time the CanSM is informed about a bus-off. When the CanSM enters the state `CANSM_BOR_CHECK_L1` or `CANSM_BOR_CHECK_L2` the Tx path is restarted so the communication should work again and the CanSM informs the application via the function `Appl_CanSM_BusOffEnd`. The according channel can be identified via the network handle, which is a parameter of both functions. The `BusOffBegin` function delivers also the number of bus-offs in a sequence. A sequence starts after the first bus-off occurred and ends when the bus-off sequence of the CAN channel had been recovered successfully. That means the CanSM enters the state `_BOR_NO_BUS_OFF` after the time `TxEnsured` exceeded.

A bus-off recovery introduced by the function `Appl_CanSM_BusOffBegin` is always “closed” by `Appl_CanSM_BusOffEnd` even if the CanSM receives a shutdown request and the bus-off recovery is interrupted.

Info
During the bus-off recovery the Nm may trigger the shutdown.

Caution
The function `Appl_CanSM_BusOffBegin` is called in interrupt context so pay attention to a SHORT runtime.

Caution
The function `Appl_CanSM_BusOffEnd` is with blocked CAN interrupts (`CANSM_EXCLUSIVE_AREA_4`) so pay attention to a SHORT runtime.

4.4 XCP notification function

The XCP gets the information if messages can be transmitted or not. I.e. the CanSM reports CANXCP_SET_ONLINE to the XCP if the state CanSM enters the state CANSM_FULL_COMMUNICATION and the passive mode is NOT activated. If CanSM leaves the state CANSM_FULL_COMMUNICATION and the passive mode is NOT activated the CanSM reports CANXCP_SET_OFFLINE to the XCP. If the CanSM is in state CANSM_FULL_COMMUNICATION and the ECU mode changes to passive CanSM reports CANXCP_SET_OFFLINE to the XCP and CANXCP_SET_ONLINE if the ECU mode switches back to active. If the CanSM is in state CANSM_FULL_COMMUNICATION and the Tx channel is deactivated caused by a bus off then the CanSM reports CANXCP_SET_OFFLINE when the CanSM enters the state CANSM_BOR_TXOFF_L1 or L2 and CANXCP_SET_ONLINE again if the state is left.

If the CanSM and the CanXCP is active at the same channel the configuration tool activates the feature by writing the XCP API function in the CanSM_Lcfg.c file. The channel assignment can be changed during link time configuration.

4.5 ECU passive mode

After the initialization of the CanSM the ECU mode is set to default value ECU active mode. The mode is the same for each CAN channel. The CanSM can be instructed to handle the passive or the active mode via the API CanSM_SetEcuPassive. The ECU mode stays until new request or a (re)initialization of the CanSM. In passive mode the CanSM sets the Tx PDU mode to OFFLINE_ACTIVE instead to ONLINE (T02). If the ECU mode switches form passive to active the CanSM switches the Tx PDUs which are in OFFLINE_ACTIVE to ONLINE.

During a bus-off recovery phase the modification of the Tx PDU mode is postponed until the bus-off recovery phase has been finished (T08, T015)

4.6 Main Function

The CanSM has one main function which has to be called cyclically by the SchM. The main function of the CanSM executes the bus-off recovery state machines (including timer and counters) of each network handle, which is configured for the CanSM. In some conditions the main function executes the network state machines too. The main function handles the Network Mode State Machine and the BOR SM for each CAN network once per cycle.

4.7 Communication Modes

The ComM collects the communication requests from the SW-C and from the network. Accordingly the ComM calculates the needed communication mode and requests this from the CAN State Manager via the function `CanSM_RequestComMode`.

4.8 Communication Mode Polling

The ComM is informed about every mode change by the CAN State Manager via the call back function `ComM_BusSM_ModeIndication`.

Additional the ComM may request the communication mode which is currently active by calling the API function `CanSM_GetCurrentComMode`. The CAN State Manager will deliver the communication mode to the pointer passed as a function parameter.

4.9 Error Handling

4.9.1 Development Error Reporting

Development errors are reported to DET using the service `Det_ReportError()`, (specified in [2]), if the pre-compile parameter `CANSM_DEV_ERROR_DETECT` is equal to `STD_ON`.

The reported CanSM ID is `0x8C`.

The reported service IDs identify the services which are described in 6.3. The following table maps the service IDs to the related services:

Service ID	Service
`0x00`	`CANSM_INIT_SERVICE_ID`
`0x01`	`CANSM_GETVERSIONINFO_SERVICE_ID`
`0x02`	`CANSM_REQUESTCOMMODE_SERVICE_ID`
`0x03`	`CANSM_GETCURRENTCOMMODE_SERVICE_ID`
`0x04`	`CANSM_CONTROLLERBUSOFF_SERVICE_ID`
`0x05`	`CANSM_MAINFUNCTION_SERVICE_ID`
`0x06`	`CANSM_PREVENTBUSSLEEPATSTARTUP_SERVICE_ID`

Table 4-3 Mapping of service IDs to services

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

Error Code		Description
`0x01`	`CANSM_E_UNINIT`	API service used without having called the initialization function.
`0x02`	`CANSM_E_PARAM_POINTER`	API service called with invalid pointer in parameter list
`0x03`	`CANSM_E_INVALID_NETWORK_HANDLE`	API service called with wrong network handle parameter, which is not configured in the CanSM configuration.
`0x04`	`CANSM_E_INVALID_NETWORK_MODE`	API service called with wrong network mode parameter
`0x05`	`CANSM_E_PARAM_CONTROLLER`	The function CanSM_ControllerBusOff is called with a wrong controller index.
`0x06`	`CANSM_E_INITLIZED`	API service used after the initialization function

Table 4-4 Errors reported to DET

4.9.1.1 Parameter Checking

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

Check Service	API called with NULL pointer invalid pointer	Network handle not exist	CanSM_Init has not been called	Requested network mode not possible	Controller Id not exist	CanSM_Init has been called
CanSM_Init	■					
CanSM_GetVersionInfo	■					
CanSM_InitMemory						
CanSM_RequestComMode			■	■		
CanSM_SetEcuPassive						
CanSM_GetCurrentComMode	■	■	■			

Check Service	API called with NULL pointer invalid pointer	Network handle not exist	CanSM_Init has not been called	Requested network mode not possible	Controller Id not exist	CanSM_Init has been called
CanSM_ControllerBusOff			■		■	
CanSM_MainFunction			■			
CanSM_PreventBusSleepAtStartUp		■				■

Table 4-5 Development Error Reporting: Assignment of checks to services

AUTOSAR requires that API functions check the validity of their parameters. The checks in Table 4-5 are internal parameter checks of the API functions. These checks are for development error reporting and can be en-/disabled via the parameter `CANSM_DEV_ERROR_DETECT`.

4.9.2 Production Code Error Reporting

Production code related errors are reported to DEM using the service `Dem_ReportErrorStatus()` (specified in [3]), if the pre-compile parameter `CANSM_PROD_ERROR_DETECT == STD_ON`. En-/disabling is an addition to the AUTOSAR standard.

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

Error Code	Description
`CANSM_E_BUSOFF_NETWORK_<X>`	The CAN State Manager reports to the DEM a network specific bus-off event each time the bus-off could be recovered or the bus-off could not be recovered within the specified tries.
`CANSM_E_MODE_CHANGE_NETWORK_<X>`	As described in 4.3.1 the CanSM repeats a transition if an error occurred. If the repetitions sequence exceed the specified `Hw Repetition` (see Table 7-1) the CanSM reports this error to the DEM.
`CANSM_E_SETTRANSCEIVERMODE_NETWORK_<X>`	If the setting of the transceiver mode was not successful the CanSM reports this error to the DEM.

Table 4-6 Errors reported to DEM

5 Integration

This chapter gives necessary information for the integration of the MICROSAR CanSM into an application environment of an ECU.

5.1 Brief Instruction

> Check if mandatory modules are available, see chapter 6.4.

> Adapt the default configuration settings of the CanSM in the configuration tool (GENy), see chapter 7.1.

> Check if the EcuM reinitializes the variables by calling `CanSM_InitMemory` before the initialization of the CanSM.

> Call `CanSM_PreventBusSleepAtStartUp` if needed (once for each network handle).

> Check if the EcuM initializes the CanSM by calling `CanSM_Init` and delivers a reference to the cluster configuration data as a function parameter.

> Check if ComM requests the desired communication mode by calling the function `CanSM_RequestComMode`. The ComM has to deliver the communication mode and the according network handle as parameters.

> Check if the SchM calls the `CanSM_MainFunction` cyclically.

5.2 Scope of Delivery

The delivery of the CanSM contains the files which are described in the chapters 5.2.1 and 5.2.2.

5.2.1 Static Files

File Name	Description
CanSM.c	This is the source file of the CanSM. It contains the implementation of the main functionality (not available if libraries are delivered).
CanSM.h	This is the main header file of the CAN state manager which is the “defines” and types of the CAN state manager.
CanSM_Cbk.h	This is the callback header file of the CAN state manager which is the specific interface of the bus-off function.
CanSM_EcuM.h	This is a header file of the CAN state manager which is the specific interface for the EcuM to the services of the CAN state manager.
CanSM_SchM.h	This is a header file of the CAN state manager which is the specific interface for the SchM to the services of the CAN state manager.
CanSM_ComM.h	This is a header file of the CAN state manager which is the specific interface for the ComM to the services of the CAN state manager.

Table 5-1 Static files

5.2.2 Dynamic Files

The dynamic files are generated by the configuration tool GENy.

File Name	Description
CanSM_Cfg.h	Configuration header file which is generated. It contains pre-compile switches, which enable/disable features, type definitions and constant values.
CanSM_Lcfg.c	Configuration source file. It contains configuration parameter which may be changed at link time.
CanSM_PBcfg.c	Configuration source file. It contains for example timer variable values or channel configuration parameter. It contains configuration parameter which may be changed after link time.

Table 5-2 Generated files

5.3 Include Structure

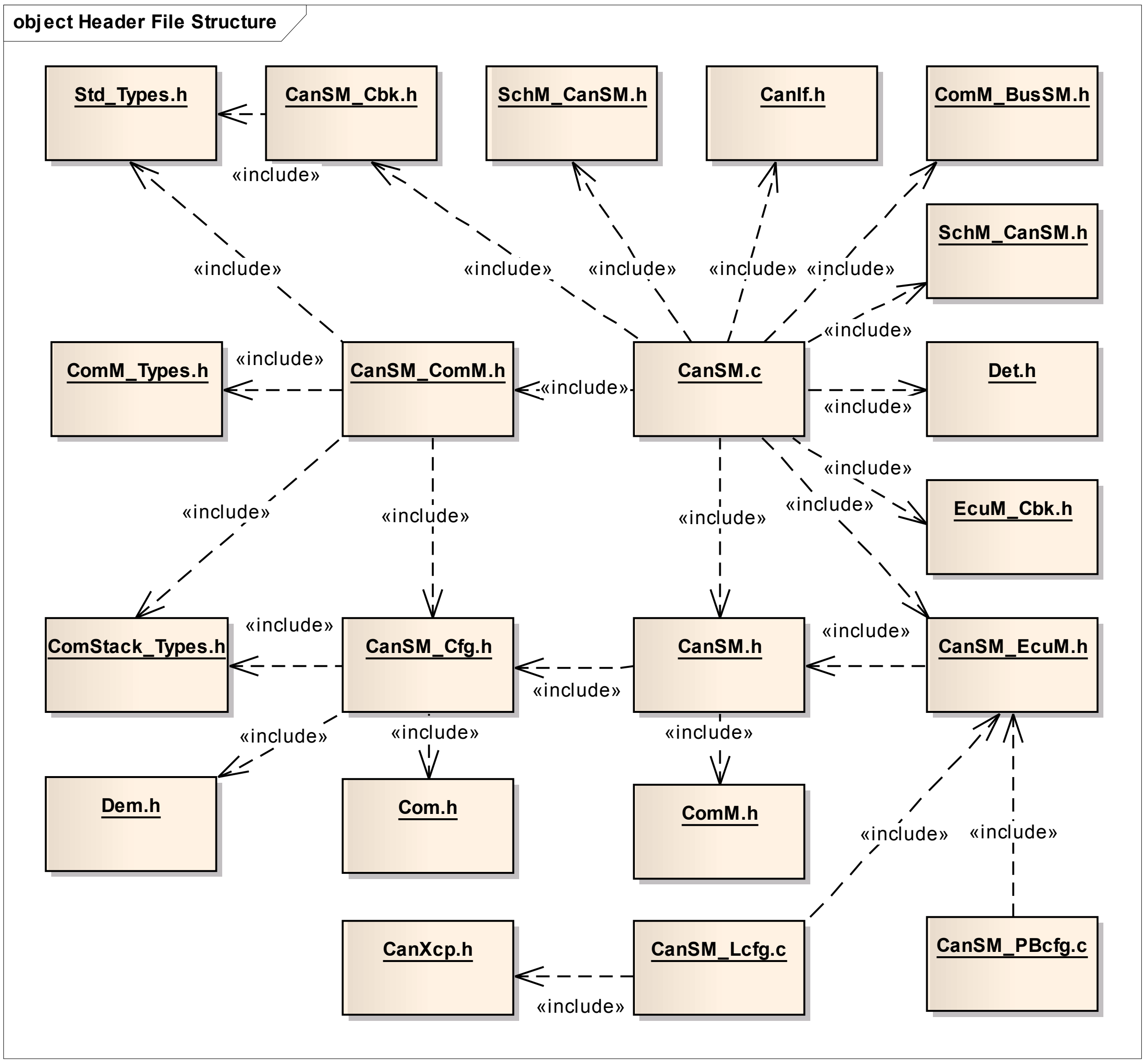


Figure 5-1 Include structure

5.4 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 names of the memory section and the compiler abstraction definitions defined for the CanSM and illustrate their mapping.

Memory Mapping Sections \ Compiler Abstraction Definitions	CANSM_VAR_NOINIT	CANSM_VAR_ZERO_INIT	CANSM_CONST	CANSM_PBCFG	CANSM_CODE
CANSM_START_SEC_VAR_NOINIT_UNSPECIFIED CANSM_STOP_SEC_VAR_NOINIT_UNSPECIFIED	■				
CANSM_START_SEC_VAR_ZERO_INIT_8BIT CANSM_STOP_SEC_VAR_ZERO_INIT_8BIT		■			
CANSM_START_SEC_VAR_NOINIT_8BIT CANSM_STOP_SEC_VAR_NOINIT_8BIT	■				
CANSM_START_SEC_CONST_8BIT CANSM_STOP_SEC_CONST_8BIT			■		
CANSM_START_SEC_CONST_32BIT CANSM_STOP_SEC_CONST_32BIT			■		
CANSM_START_SEC_CONST_UNSPECIFIED CANSM_STOP_SEC_CONST_UNSPECIFIED			■		
CANSM_START_SEC_PBCFG CANSM_STOP_SEC_PBCFG				■	
CANSM_START_SEC_CODE CANSM_STOP_SEC_CODE					■

Table 5-3 Compiler abstraction and memory mapping

5.5 Critical Areas

The AUTOSAR standard provides with the BSW Scheduler a BSW module, which handles entering and leaving critical sections. The Vector AUTOSAR CanSM supports additional solutions to handle these sections.

Critical sections are supported by the following three alternatives:

> the BSW Scheduler

> the Vector Standard Library (VStdLib)

> the proprietary CAN interrupts locking functions

The VStdLib offers the possibility of mapping the interrupt handling to OS services, or to user defined functions. In the first case interrupt handling is done by the OS, in the second case the user has to take care by providing corresponding functions.

The intention of the following critical sections is to block the interrupt of CanSM functions (with a higher priority).

> The `CANSM_EXCLUSIVE_AREA_1` has to be used if it's possible that the function `CanSM_MainFunction()` may be interrupted by any of the functions `CanSM_RequestComMode()` or `CanSM_SetEcuPassive()`.

> The `CANSM_EXCLUSIVE_AREA_2` has to be used if it's possible that the function `CanSM_RequestComMode()` may be interrupted by any of the functions `CanSM_MainFunction()` or `CanSM_SetEcuPassive()`.

> The `CANSM_EXCLUSIVE_AREA_3` has to be used if it's possible that the function `CanSM_SetEcuPassive()` may be interrupted by any of the functions `CanSM_RequestComMode()` or `CanSM_MainFunction ()`.

The intention of the following critical sections is to avoid a change of the CAN controller or transceiver mode during shutdown of the CAN communication when the CanSM performs the transition to from SilentCommunication to NoCommunication.

> The `CANSM_EXCLUSIVE_AREA_4` has to be used if it's possible that one of functions `CanSM_MainFunction()` or `CanSM_RequestComMode()` may be interrupted by a CAN event.

1. By Can Wake Up Interrupt
2. By Can Wake Up Polling
3. By Can bus-off (Can error)

> The `CANSM_EXCLUSIVE_AREA_5` has to be used if it's possible that one of functions `CanSM_MainFunction()` or `CanSM_RequestComMode()` may be interrupted by a CAN wakeup:

1. By Can Wake Up Interrupt
2. By Can Wake Up Polling
3. By Wakeup by transceiver

6 API Description

6.1 Interfaces Overview

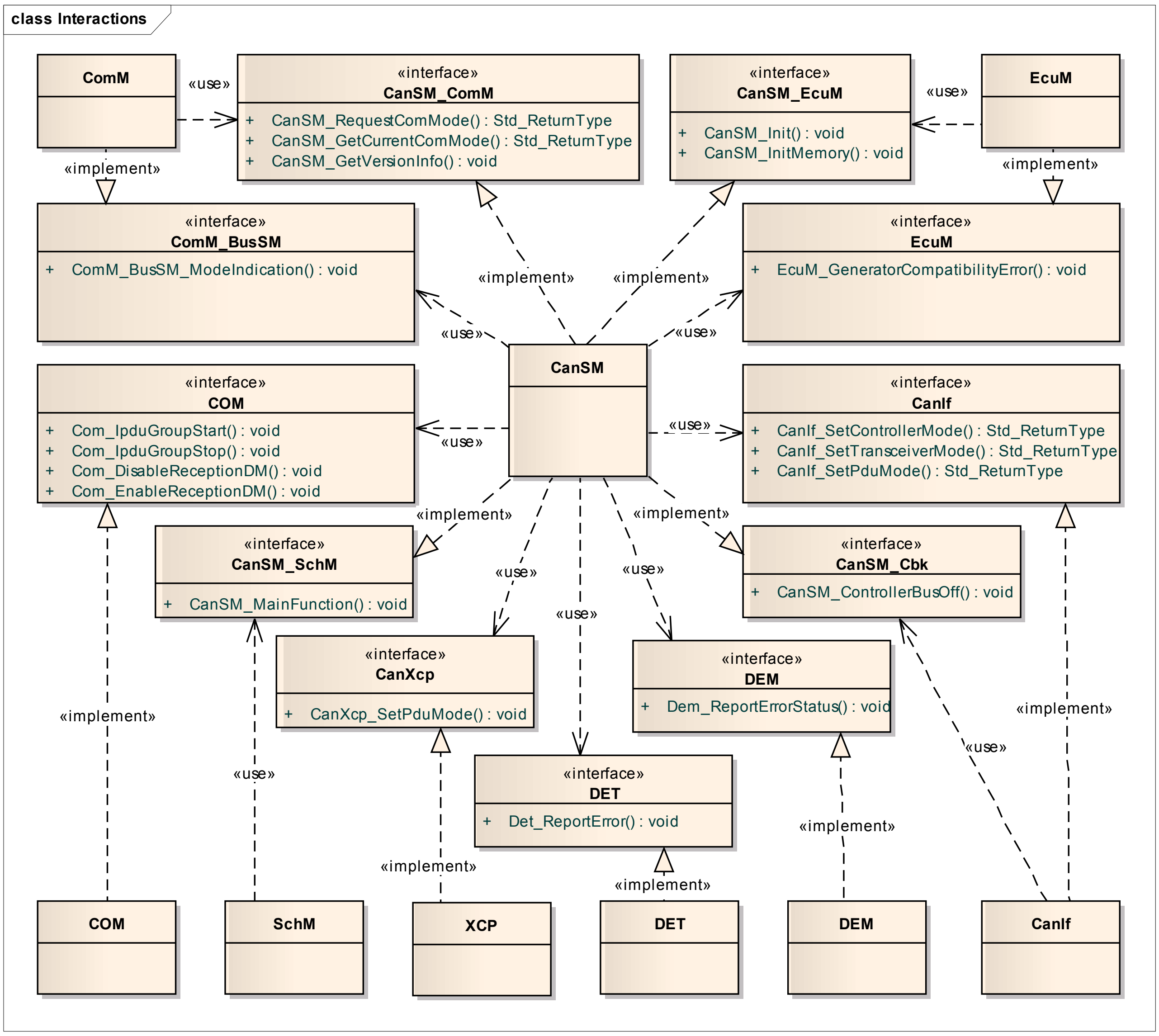

class Interactions
ComM
«use»
«interface» CanSM_ComM
+ CanSM_RequestComMode() : Std_ReturnType
+ CanSM_GetCurrentComMode() : Std_ReturnType
+ CanSM_GetVersionInfo() : void
«interface» CanSM_EcuM
+ CanSM_Init() : void
+ CanSM_InitMemory() : void
EcuM
«implement»
«interface» ComM_BusSM
+ ComM_BusSM_ModeIndication() : void
«interface» EcuM
+ EcuM_GeneratorCompatibilityError() : void
CanSM
«interface» COM
+ Com_IpduGroupStart() : void
+ Com_IpduGroupStop() : void
+ Com_DisableReceptionDM() : void
+ Com_EnableReceptionDM() : void
«interface» CanIf
+ CanIf_SetControllerMode() : Std_ReturnType
+ CanIf_SetTransceiverMode() : Std_ReturnType
+ CanIf_SetPduMode() : Std_ReturnType
«interface» CanSM_SchM
+ CanSM_MainFunction() : void
«interface» CanSM_Cbk
+ CanSM_ControllerBusOff() : void
«interface» CanXcp
+ CanXcp_SetPduMode() : void
«interface» DEM
+ Dem_ReportErrorStatus() : void
«interface» DET
+ Det_ReportError() : void
COM
SchM
XCP
DET
DEM
CanIf

Figure 6-1 CanSM interactions with other BSW

6.2 Type Definitions

Type Name	C-Type	Description	Value Range

CanSM_NetworkModeStateType	uint8	This type defines the states of the network mode state machine.	`CANSM_UNINITED`
			CanSM is not initialized in this state
			`CANSM_NO_COMMUNICATION`

Type Name	C-Type	Description	Value Range
			ComM requests `COMM_NO_COMMUNICATION` in this state
			`CANSM_SILENT_COMMUNICATION` ComM requests `COMM_SILENT_COMMUNICATION` in this state
			`CANSM_FULL_COMMUNICATION` ComM requests `COMM_FULL_COMMUNICATION` in this state
CanSM_BusOffRecoveryStateType	uint8	This type defines the states of the bus-off recovery state machine.	`CANSM_BOR_IDLE` Idle state
			`CANSM_BOR_CHECK` Initial bus-off check at beginning of full-communication
			`CANSM_BOR_NO_BUS_OFF` Regular state during full-communication without detected bus-off events
			`CANSM_BOR_TXOFF_L1` Bus-off recovery level 1 state, TX disabled
			`CANSM_BOR_CHECK_L1` Bus-off recovery level 1 state, TX enabled again
			`CANSM_BOR_TXOFF_L2` Bus-off recovery level 2 state, TX disabled
			`CANSM_BOR_CHECK_L2` Bus-off recovery level 2 state, TX enabled again

Table 6-1 Type definitions

6.3 Services Provided by CanSM

The CanSM API consists of services, which are realized by function calls.

Return value	Function name	Function parameter
`void`	`CanSM_Init`	CanSM_ConfigType * ConfigPtr XOR void
`void`	`CanSM_GetVersionInfo`	Std_VersionInfoType VersionInfo
`Std_ReturnType`	`CanSM_RequestComMode`	NetworkHandleType NetworkHandle, ComM_ModeType ComM_Mode

Void	CanSM_SetEcuPassive	Boolean CanSM_EcuPassiveMode
Std_ReturnType	CanSM_GetCurrentComMode	NetworkHandleType NetworkHandle, ComM_ModeType* ComM_ModePtr
void	CanSM_ControllerBusOff	uint8 Controller
void	CanSM_MainFunction	void
Std_ReturnType	CanSM_PreventBusSleepAtStartUp	NetworkHandleType NetworkHandle

Table 6-2 API Overview

6.3.1 CanSM_InitMemory

Prototype	
`void CanSM_InitMemory( void )`	
Parameter	
-	-
Return code	
-	-
Functional Description	
This function initializes the CanSM memory and sets the variable `CanSm_IsInitialized` to `FALSE`	
Particularities and Limitations	
■ Called once at start-up before the initialization function.	
Expected Caller Context	
■ Function is called once before CanSM_Init	

Table 6-3 CanSM_InitMemory

6.3.2 CanSM_Init

Prototype	
VARIANT-PRE-COMPILE, VARIANT-LINK-TIME	`void CanSM_Init( void )`
VARIANT-POST-BUILD	`void CanSM_Init( const CanSM_ConfigType* ConfigPtr )`
Parameter	
`ConfigPtr`	Pointer to the configuration structure that shall be used for the post-build parameters.
Return code	
-	-
Functional Description	
Service for CAN station manager initialization. If the feature “Multiple Identity” is activated the function requires the configuration pointer independent of the chosen configuration variant.	
Particularities and Limitations	
■ Called once at start-up by ECU state manager after the memory initialization function. ■ Non Reentrant	

Expected Caller Context
■ Called once after startup

Table 6-4 CanSM_Init

6.3.3 CanSM_MainFunction

Prototype	
`void CanSM_MainFunction( void );`	
Parameter	
-	-
Return code	
-	-
Functional Description	
The main function of the CanSM executes the bus-off recovery state machines (including timer and counters) of each network handle, which is configured for the CanSM. The main function executes the network state machine too.	
Particularities and Limitations	
■ CanSM has to be initialized. Function has to be called cyclic. The cycle time is set in the configuration tool. ■ Non Reentrant	
Expected Caller Context	
■ Cyclic on task level	

Table 6-5 CanSM_MainFunction

6.3.4 CanSM_RequestComMode

Prototype	
`Std_ReturnType CanSM_RequestComMode( NetworkHandleType NetworkHandle, ComM_ModeType ComM_Mode )`	
Parameter	
`NetworkHandle`	The communication network number belonging to the request.
`ComM_Mode`	New desired value of the communication mode.
Return code	
`ReturnType`	Returns whether function parameter are valid or not.
Functional Description	
The function stores the requested communication mode for the network handle and executes the corresponding network mode state machine.	
Particularities and Limitations	
■ CanSM has to be initialized. ■ Reentrant for different CAN networks, not reentrant for the same CAN network	
Expected Caller Context	
■ Function can be called in task and interrupt context.	

Table 6-6 CanSM_RequestComMode

6.3.5 CanSM_SetEcuPassive

Prototype	
`void CanSM_SetEcuPassive( boolean CanSM_EcuPassiveMode )`	
Parameter	
`CanSM_EcuPassiveMode`	Boolean parameter which switches the ECU mode between active and passive mode
Return code	
-	-
Functional Description	
The function stores the requested ECU mode until it's modified by the next call of this function. In passive mode the CanSM sets the Tx PDU mode to OFFLINE_ACTIVE instead to ONLINE.	
Particularities and Limitations	
■ CanSM has to be initialized.	
Expected Caller Context	
■ Function can be called in task and interrupt context.	

Table 6-7 CanSM_SetEcuPassive

6.3.6 CanSM_GetCurrentComMode

Prototype	
`Std_ReturnType CanSM_GetCurrentComMode( NetworkHandleType NetworkHandle, ComM_ModeType* ComM_ModePtr )`	
Parameter	
`NetworkHandle`	Index of the network channel.
`ComM_ModePtr`	Pointer where the communication mode information is copied to.
Return code	
`ReturnType`	Returns whether function parameter are valid or not.
Functional Description	
This service delivers the current communication mode of a CAN network.	
Particularities and Limitations	
■ CanSM has to be initialized.	
Expected Caller Context	
■ Function can be called in task and interrupt context.	

Table 6-8 CanSM_GetCurrentComMode

6.3.7 CanSM_GetVersionInfo

Prototype	
`void CanSM_GetVersionInfo( Std_VersionInfoType * VersionInfo )`	
Parameter	
`VersionInfo`	Pointer, where to store the version data of the CanSM.
Return code	
-	-

Functional Description
This service returns the version information of this module. The version information includes: - Module Id - Vendor Id - Vendor specific version numbers. This function shall be pre compile time configurable On/Off by the configuration parameter: CANSM_VERSION_INFO_API
Particularities and Limitations
■ The function is only available if enabled at compile time (CANSM_VERSION_INFO_API = STD_ON)
Expected Caller Context
■ Function can be called in task and interrupt context.

Table 6-9 CanSM_GetVersionInfo

6.3.8 CanSM_PreventBusSleepAtStartUp

Prototype	
`Std_ReturnType CanSM_PreventBusSleepAtStartUp( NetworkHandleType CanSM_NetworkHandle )`	
Parameter	
`NetworkHandleType`	communication network handle
Return code	
`Std_ReturnType`	Returns whether the network handle is valid and if the function has been called before or after the initialization.
Functional Description	
The function can be used to prevent the bus sleep state of the CanIf, CanDrv and CanTrcv at start up for the given CAN network handle. The CanIf, CanDrv and CanTrcv leaves in the corresponding module initialization state. For further information refer to [7].	
Particularities and Limitations	
■ Called at start-up before the CanSM initialization function	
Expected Caller Context	
■ Function has to be called before CanSM_Init	

Table 6-10 CanSM_PreventBusSleepAtStartUp

6.4 Services used by CanSM

The following services are provided by other components. The services are used by the CanSM. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
CanIf	CanIf_SetControllerMode
CanIf	CanIf_GetControllerMode
CanIf	CanIf_SetTransceiverMode
CanIf	CanIf_GetTransceiverMode

Component	API
CanIf	CanIf_SetPduMode
DEM	Dem_ReportErrorStatus
DET	Det_ReportError
EcuM	EcuM_GeneratorCompatibilityError (see [8])
ComM	ComM_BusSM_ModeIndication
Com	Com_IpduGroupStart
Com	Com_IpduGroupStop
Com	Com_DisableReceptionDM
Com	Com_EnableReceptionDM
SchM	SchM_Enter_CanSM(ExclusiveArea)
SchM	SchM_Exit_CanSM(ExclusiveArea)
XCP	CanXcp_SetPduMode

Table 6-11 Services used by the CanSM

6.5 Callback Functions

This chapter describes the callback functions that are implemented by the CanSM and can be invoked by other modules. The prototypes of the callback functions are provided in the header file CanSM_Cbk.h by the CanSM.

6.5.1 CanSM_ControllerBusOff

Prototype	
`void CanSM_ControllerBusOff( uint8 Controller )`	
Parameter	
`Controller`	Index of the CAN controller, which detected a bus-off event
Return code	
-	-
Functional Description	
The CanSM is notified about a bus-off event on a certain CAN controller with this callback function. The CanSM uses this information to execute the bus-off recovery for the corresponding controller.	
Particularities and Limitations	
■ CanSM has to be initialized.	
Expected Caller Context	
■ Function can be called in task and interrupt context.	

Table 6-12 CanSM_ControllerBusOff

6.5.2 Appl_CanSM_BusOffBegin

Prototype
`void Appl_CanSM_BusOffBegin( NetworkHandleType CanSm_NetworkHandle, CanSM_BorCounterType CanSM_BusOffNotificationCounter )`

Parameter	
NetworkHandleType	communication network handle
CanSM_BusOffNotificationCounter	Number of bus-offs in a sequence
Return code	
–	-
Functional Description	
The function informs the application about a occurred bus-off	
Particularities and Limitations	
■	
Expected Caller Context	
■ Function can be called in task and interrupt context.	

Table 6-13 Appl_CanSM_BusOffBegin

6.5.3 Appl_CanSM_BusOffEnd

Prototype	
void Appl_CanSM_BusOffEnd (NetworkHandleType CanSm_NetworkHandle)	
Parameter	
NetworkHandleType	communication network handle
Return code	
–	-
Functional Description	
The function informs the application if the Tx communication is restarted after a bus-off event.	
Particularities and Limitations	
■	
Expected Caller Context	
■ Function is called in CanSM task context.	

Table 6-14 Appl_CanSM_BusOffEnd

7 Configuration

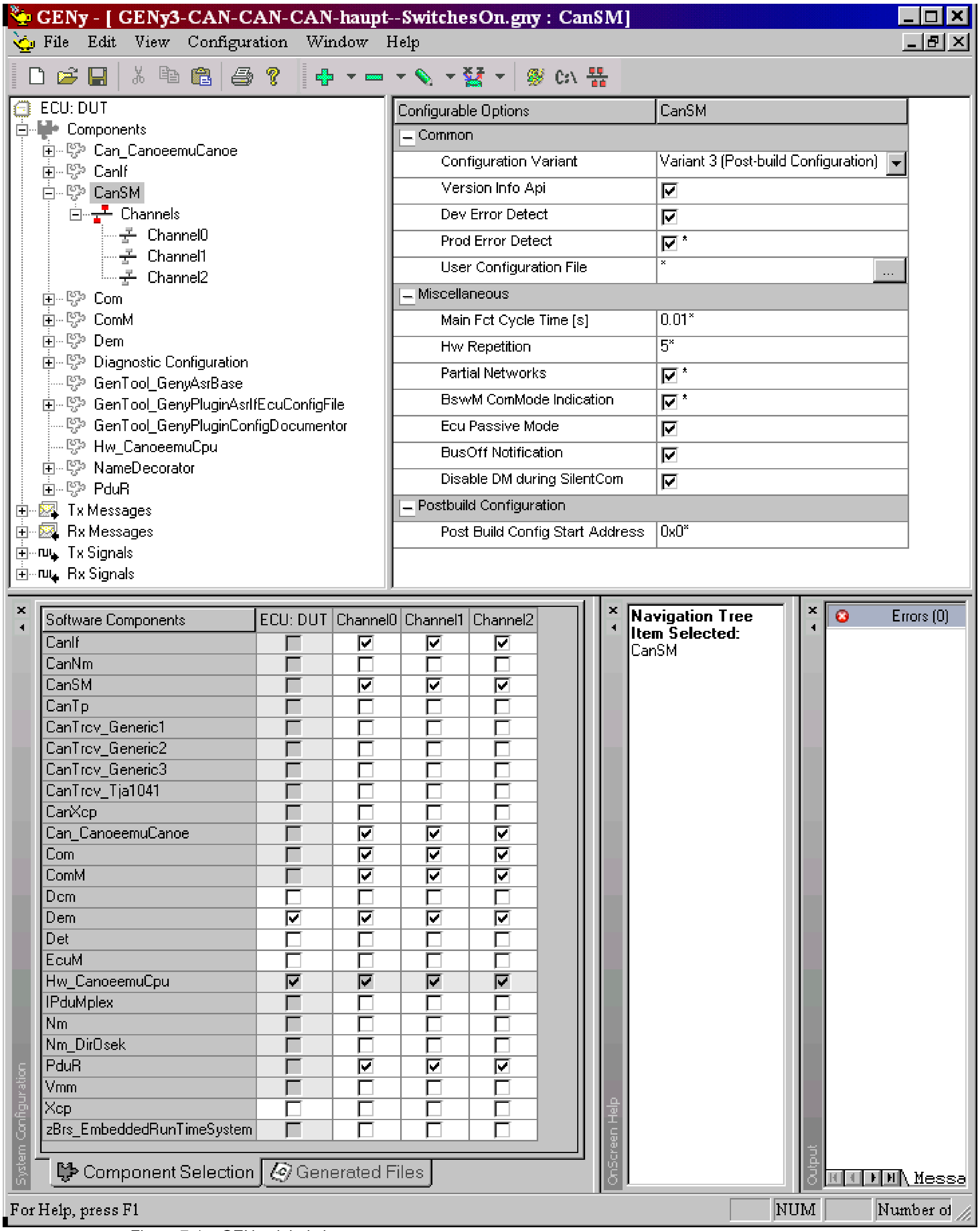
The attributes of the CanSM can be configured with the following methods:

> Configuration in GENy for a detailed description see 7.1

7.1 Configuration with GENy

7.1.1.1 Basic

The CanSM is configured with the help of the configuration tool GENy. Settings for the CanSM can be selected in the GUI, if you choose `Components, CanSM` in the tree at the left. These settings are used to generate the configuration header which is needed to compile the component and the code files which may be linked to the object code.



Configurable Options	CanSM
Common	
Configuration Variant	Variant 3 (Post-build Configuration)
Version Info Api	✓
Dev Error Detect	✓
Prod Error Detect	✓ *
User Configuration File	*
Miscellaneous	
Main Fct Cycle Time [s]	0.01*
Hw Repetition	5*
Partial Networks	✓ *
BswM ComMode Indication	✓ *
Ecu Passive Mode	✓
BusOff Notification	✓
Disable DM during SilentCom	✓
Postbuild Configuration	
Post Build Config Start Address	0x0*

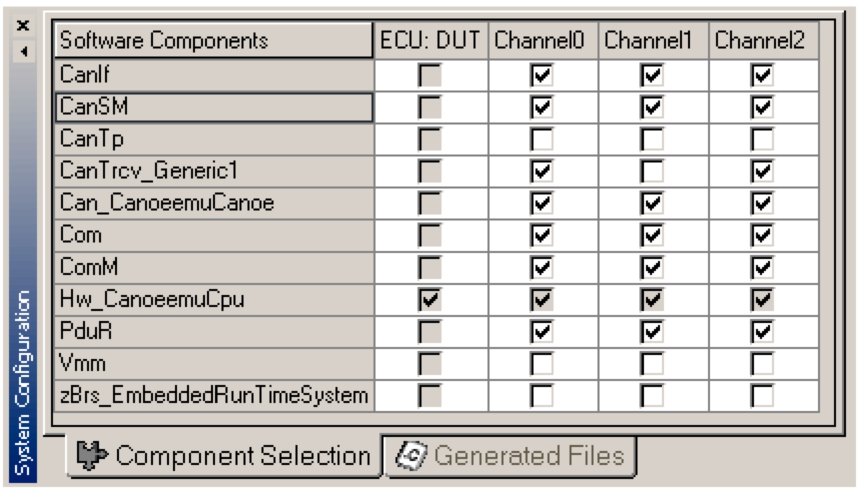
Software Components	ECU: DUT	Channel0	Channel1	Channel2
CanIf	☐	✓	✓	✓
CanNm	☐	☐	☐	☐
CanSM	☐	✓	✓	✓
CanTp	☐	☐	☐	☐
CanTrcv_Generic1	☐	☐	☐	☐
CanTrcv_Generic2	☐	☐	☐	☐
CanTrcv_Generic3	☐	☐	☐	☐
CanTrcv_Tja1041	☐	☐	☐	☐
CanXcp	☐	☐	☐	☐
Can_CanoeemuCanoe	☐	✓	✓	✓
Com	☐	✓	✓	✓
ComM	☐	✓	✓	✓
Dcm	☐	☐	☐	☐
Dem	✓	✓	✓	✓
Det	☐	☐	☐	☐
EcuM	☐	☐	☐	☐
Hw_CanoeemuCpu	✓	✓	✓	✓
IPduMplex	☐	☐	☐	☐
Nm	☐	☐	☐	☐
Nm_DirOsek	☐	☐	☐	☐
PduR	☐	✓	✓	✓
Vmm	☐	☐	☐	☐
Xcp	☐	☐	☐	☐
zBrs_EmbeddedRunTimeSystem	☐	☐	☐	☐

Navigation Tree Item Selected:
CanSM

Figure 7-1 GENy global view

7.1.2 Activation of the CAN State Manager

If the CanSM isn’t available in the navigation tree you have to assign the desired channels to the CanSM by checking the corresponding box in the tab Component Selection of the System Configuration, see Figure 7-2. If no CAN channel is available add one via “Configuration”, “Add Channel”.



Software Components	ECU: DUT	Channel0	Channel1	Channel2
CanIf	☐	✓	✓	✓
CanSM	☐	✓	✓	✓
CanTp	☐	☐	☐	☐
CanTrcv_Generic1	☐	✓	☐	✓
Can_CanoeemuCanoe	☐	✓	✓	✓
Com	☐	✓	✓	✓
ComM	☐	✓	✓	✓
Hw_CanoeemuCpu	✓	✓	✓	✓
PduR	☐	✓	✓	✓
Vmm	☐	☐	☐	☐
zBrs_EmbeddedRunTimeSystem	☐	☐	☐	☐

Figure 7-2 GENy channel selection

Pay also attention that the Com and the ComM are activated in each channel which has an active CanSM. The Com may only selectable if the channel has an activated driver.

Info
The detailed information for all checkboxes and settings is given in the so-called OnScreen Help view of GENy just by clicking the checkbox or the name of the switch. This is activated via **View | OnScreen Help**.

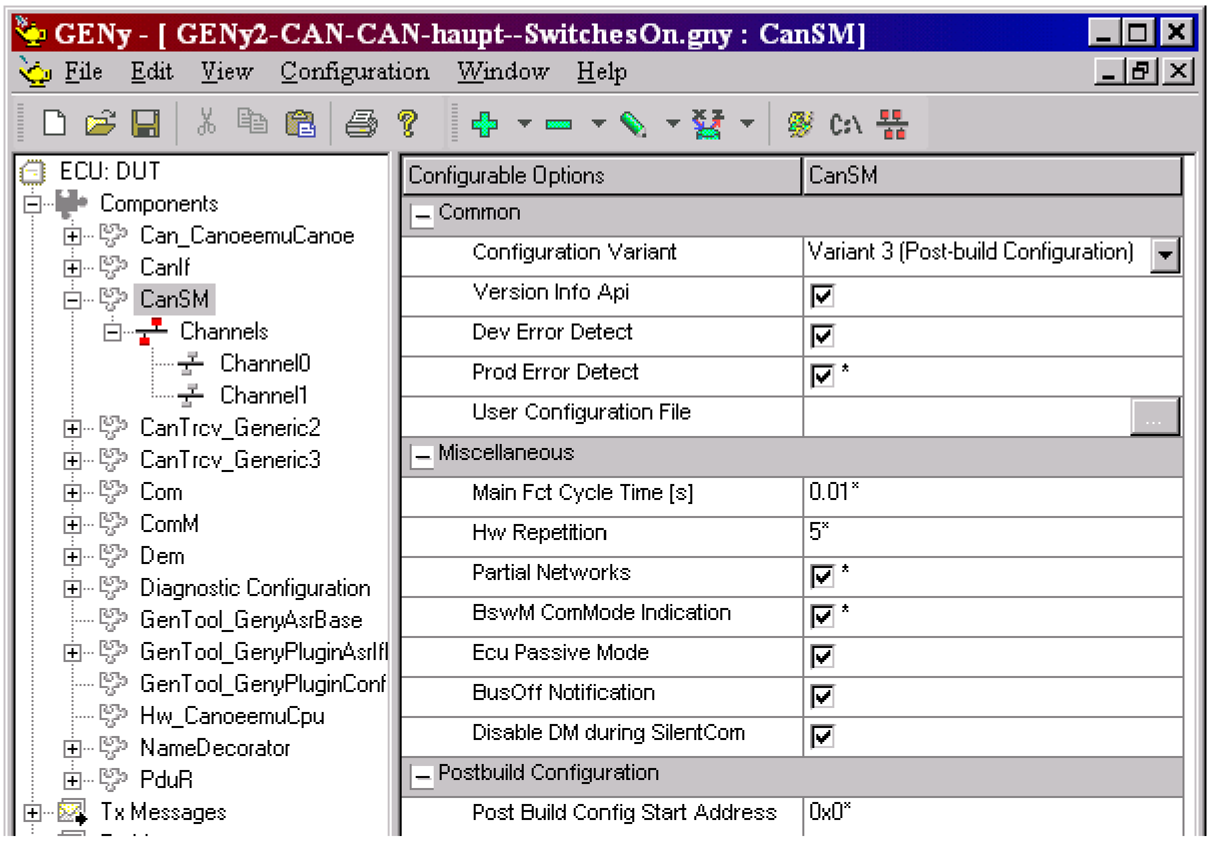
Information about how to work with GENy can be found in the “Online Help” of GENy opened via **Help | Help topics**.

7.1.3 General Settings

The main function of the CanSM is called once each cycle time by the SchM. The cycle time is defined at configuration time and shall not be changed during runtime because this functionality requires a fixed timing. If you choose a lower cycle time the performance of the CanSM state machines will increase a little and the chosen thresholds of the timers will have a smaller deviation. On the other hand more calls of the main function will be wasteful, if the CanSM has executed all jobs and the performance of the overall system may be reduced. Enter the desired time in the line Main Fct Cycle Time, see Figure 7-3.

Caution
The resolution of the timer is limited by the cycle time. Therefore is the duration of the timers a multiple of the cycle time. Nevertheless the desired time can be entered direct in seconds and the tool will convert it in the fitting numbers of cycle times. Due to this conversion you may have a deviation of a cycle time.

With the “switches” of the CanSM configuration you can activate or deactivate functionalities to adapt the CanSM to your needs.



ECU: DUT
- Components
  - Can_CanoeemuCanoe
  - CanIf
  - CanSM
    - Channels
      - Channel0
      - Channel1
  - CanTrcv_Generic2
  - CanTrcv_Generic3
  - Com
  - ComM
  - Dem
  - Diagnostic Configuration
  - GenTool_GenyAsrBase
  - GenTool_GenyPluginAsrIf
  - GenTool_GenyPluginConf
  - Hw_CanoeemuCpu
  - NameDecorator
  - PduR
- Tx Messages

Configurable Options	CanSM
Common	
Configuration Variant	Variant 3 (Post-build Configuration)
Version Info Api	✓
Dev Error Detect	✓
Prod Error Detect	✓ *
User Configuration File	...
Miscellaneous	
Main Fct Cycle Time [s]	0.01*
Hw Repetition	5*
Partial Networks	✓ *
BswM ComMode Indication	✓ *
Ecu Passive Mode	✓
BusOff Notification	✓
Disable DM during SilentCom	✓
Postbuild Configuration	
Post Build Config Start Address	0x0*

Figure 7-3 GENy setup switches

Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
Common				
`Configuration Variant`	PRE-COMPILE	define	**1**, 2, 3	The CAN State Manager supports pre-compile (1), link time (2) and post build (3) configuration. The generation tool provides the possibility to choose the kind of configuration.
`Version Info Api`	PRE-COMPILE	define	STD_ON **STD_OFF**	The CAN State Manager allows to enabling and disabling the version info API function.
`Dev Error Detect`	PRE-COMPILE	define	STD_ON **STD_OFF**	The CAN State Manager allows enabling and disabling the development error notification mechanism. If the checkbox isn’t selected the CanSM will remove the function calls and the header includes of the DET.
`Prod Error Detect`	PRE-COMPILE	define	STD_ON **STD_OFF**	The CanSM allows activating and deactivating the production error messages to the DEM.

Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
`User Configuration File`	PRE-COMPILE	define	**-**	If a file path will be specified, the file is read and its contents are put at the end of the CanSM configuration file.
Miscellaneous				
`Main Fct Cycle Time [s]`	PRE-COMPILE	-	**0.0100**	Specifies the time base of the CanSM. The main function of the CanSM is called once each cycle time by the Schedule Manager.
`Hw Repetition`	PRE-COMPILE, LINK-TIME	uint8	**5**	The value specifies how often the CanSM tries to recover a failed transition. If the number is exceeded the CanSM triggers a DEM failed message (see 4.3.1).
`Partial Networks`	PRE-COMPILE	define	STD_ON **STD_OFF**	If enabled the CanSM changes the ASR shutdown sequence according to the needs of Partial Network Transceivers
`BswM ComMode Indication`	PRE-COMPILE	define	STD_ON **STD_OFF**	If enabled the CanSM inform the BswM about the communication mode via BswM_CanSM_CurrentState()
`Ecu Passive Mode`	PRE-COMPILE	define	STD_ON **STD_OFF**	En-/ disable the ECU Passive Mode handling. (De)Activate the feature if it is (de)activated in the ComM too. In passive mode the CanSM sets the Tx PDU mode to OFFLINE_ACTIVE instead to ONLINE.
`BusOff Notification`	PRE-COMPILE	define	STD_ON **STD_OFF**	En-/ disable the bus-off notification functions `Appl_CanSM_BusOffBegin` and `Appl_CanSM_BusOffEnd`. The application is informed about each bus-off (Begin) and if the CAN channel is restarted (End).
`Disable DM during SilentCom`	PRE-COMPILE	define	STD_ON **STD_OFF**	If enabled the CanSM disables the deadline monitoring of the RX PDU groups in transition to `CANSM_SILENT_COMMUNICATION`.
Postbuild Configuration				
Post Build Config Start Address	PRE-COMPILE, LINK-TIME	address	0x0	Set the Start Address of the module's post-build configuration. The Modul Start Address must specify the start address of the memory area, where the generated hex-file will be mapped to. The attribute is only evaluated and visible if the configuration variant is set to Variant 3 (Post-build configuration).

Table 7-1 General configuration Settings in GENy

Info
The transceiver handling will be activated automatic if a transceiver is activated in the `Component Selection`. In Figure 7-2 the transceiver handling is activated at the first and the last channel.

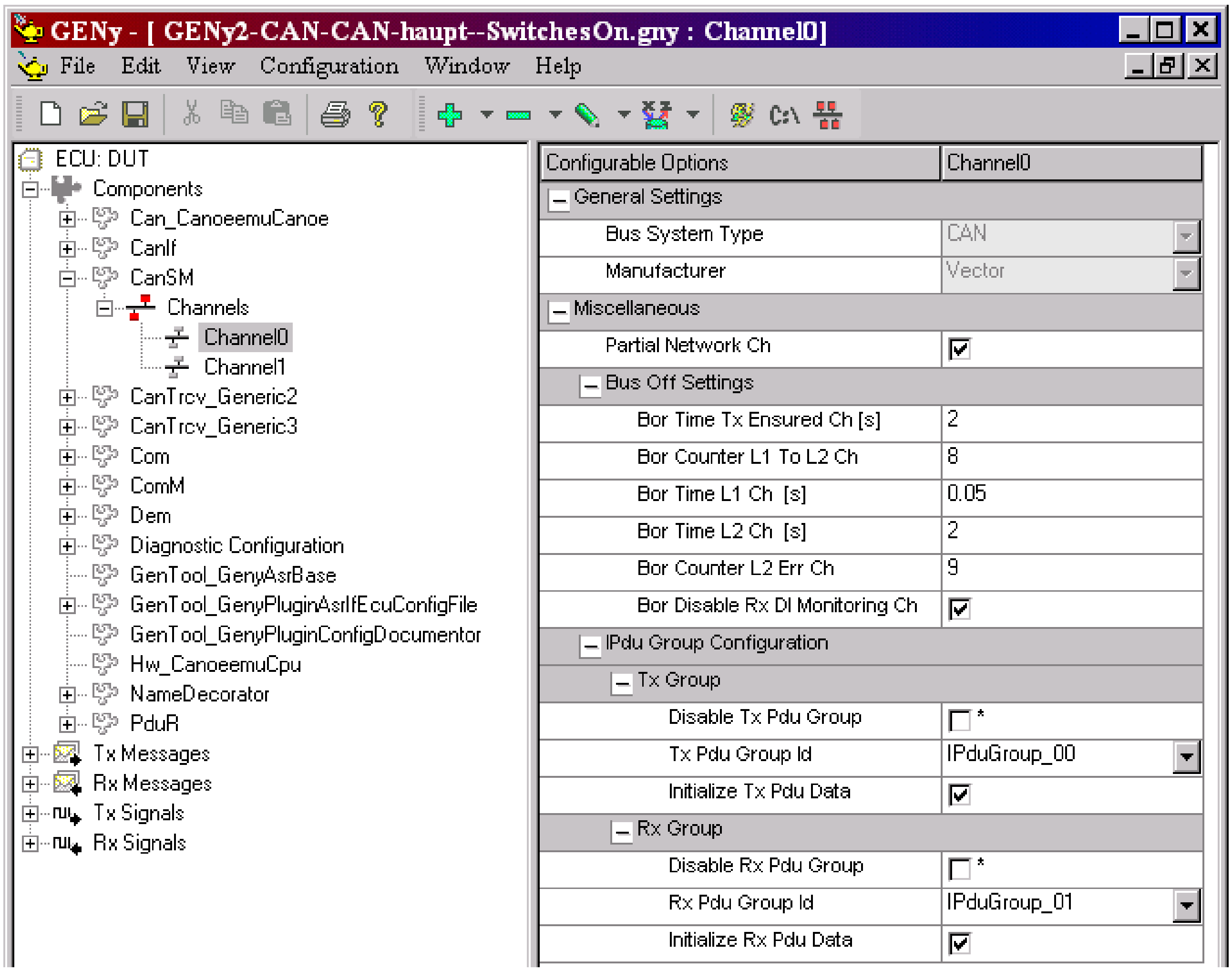
The CanSM checks if the CanIf_SetTransceiverMode was successful. If the CAN network doesn't start check which transceiver mode is returned by the function CanIf_GetTransceiverMode. The returned transceiver mode has to be equal to the mode which was requested by the CanSM before.

7.1.4 Network Settings

The number of networks may be changed at link time. The user doesn't need to enter the number of networks manually. The configuration tool calculates this value automatically.

7.1.5 General CAN Channel Specific Settings

The parameter described in this chapter can be set individual for each CAN channel.



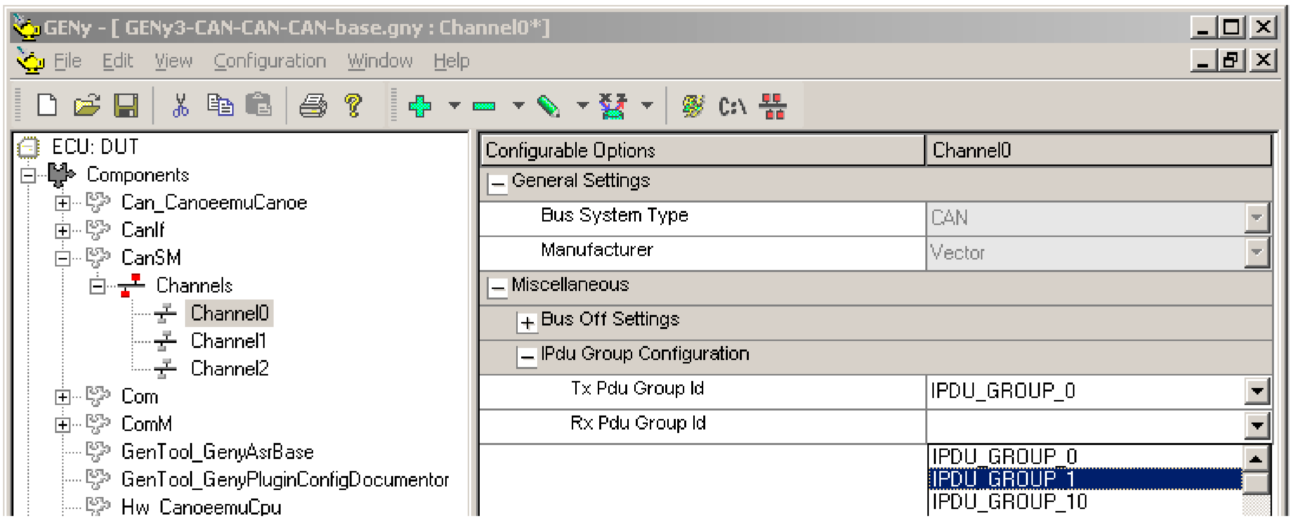
Configurable Options	Channel0
General Settings	
Bus System Type	CAN
Manufacturer	Vector
Miscellaneous	
Partial Network Ch	✓
Bus Off Settings	
Bor Time Tx Ensured Ch [s]	2
Bor Counter L1 To L2 Ch	8
Bor Time L1 Ch [s]	0.05
Bor Time L2 Ch [s]	2
Bor Counter L2 Err Ch	9
Bor Disable Rx Dl Monitoring Ch	✓
IPdu Group Configuration	
Tx Group	
Disable Tx Pdu Group	☐ *
Tx Pdu Group Id	IPduGroup_00
Initialize Tx Pdu Data	✓
Rx Group	
Disable Rx Pdu Group	☐ *
Rx Pdu Group Id	IPduGroup_01
Initialize Rx Pdu Data	✓

Figure 7-4 GENy CAN channel specific configuration

Each CAN network needs the variable names of the PDU groups. Select the desired PDU group name in the dropdown menu of the CAN channel, see Figure 7-5.

Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
Miscellaneous				
`Partial Network Ch`	PRE-COMPILE, LINK-TIME, POST-BUILD	boolean	**false**	If Enabled then the CanSM changes the shutdown sequence to the needs of the partial network feature. The global switch has to be enabled, too.

Table 7-2 Partial Network



GENy - [GENy3-CAN-CAN-CAN-base.gny : Channel0*]

ECU: DUT
- Components
  - Can_CanoeemuCanoe
  - CanIf
  - CanSM
    - Channels
      - Channel0
      - Channel1
      - Channel2
  - Com
  - ComM
  - GenTool_GenyAsrBase
  - GenTool_GenyPluginConfigDocumentor
  - Hw_CanoeemuCpu

Configurable Options	Channel0
General Settings	
Bus System Type	CAN
Manufacturer	Vector
Miscellaneous	
Bus Off Settings	
IPdu Group Configuration	
Tx Pdu Group Id	IPDU_GROUP_0
Rx Pdu Group Id	

IPDU_GROUP_0
IPDU_GROUP_1
IPDU_GROUP_10

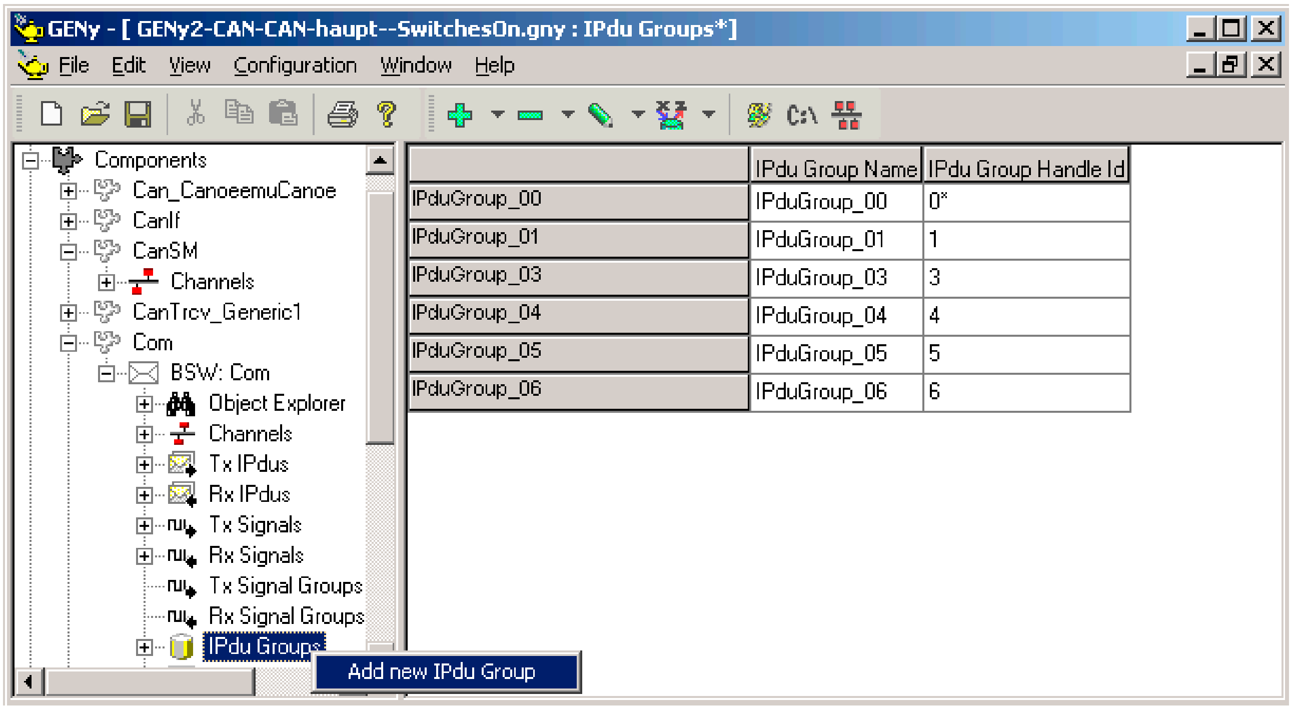
Figure 7-5 GENy CAN channel specific PDU group configuration

Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
IPDU groups				
`Disable Tx Pdu Group`	PRE-COMPILE, LINK-TIME	boolean	**false**	If Enabled then the CanSM disables the TX PDU groups handling for this channel. Activate the option if the channel has no IPDU group.
`Tx PDU Group Id`	PRE-COMPILE, LINK-TIME	Name	-	Identifier to connect variable used by CanSM to the desired PDU group of the Com.
`Initialize Tx Pdu Data`	PRE-COMPILE, LINK-TIME	boolean	**false**	If Enabled then the CanSM requests the initialization of the data in the I-PDUs of this I-PDU group (see second parameter of the function `Com_IpduGroupStart`).
`Disable Rx Pdu Group`	PRE-COMPILE, LINK-TIME	boolean	**false**	If Enabled then the CanSM disables the TX PDU groups handling for this channel. Activate the option if the channel has no IPDU group.
`Rx PDU Group Id`	PRE-COMPILE, LINK-TIME	name	-	Identifier to connect variable used by CanSM to the desired PDU group of the Com.
`Initialize Rx`	PRE-COMPILE,	boolean	**false**	If Enabled then the CanSM requests

Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
`Pdu Data`	LINK-TIME			the initialization of the data in the I-PDUs of this I-PDU group (see second parameter of the function `Com_IpduGroupStart`).

Table 7-3 Channel configuration

If the list is empty check if the Com module is activated in the component selection, see Figure 7-2. If the Com is available and the list is still empty it's also possible that the PDU list in the Com is empty. Add new PDUs by a click with the right mouse button on the IPdu Groups, see Figure 7-6.



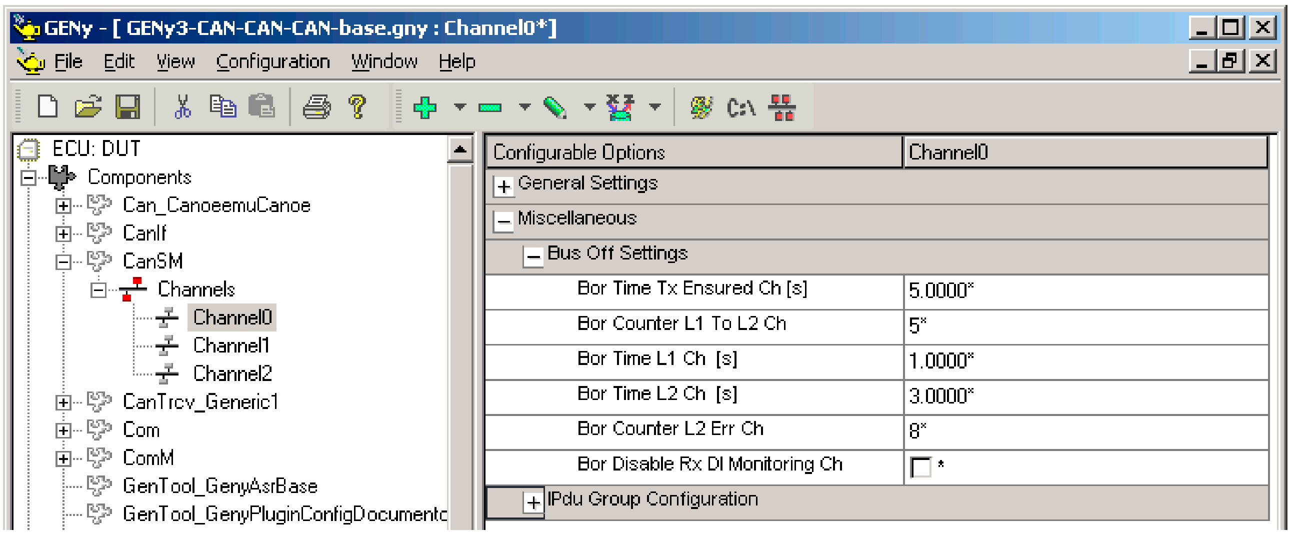
	IPdu Group Name	IPdu Group Handle Id
IPduGroup_00	IPduGroup_00	0*
IPduGroup_01	IPduGroup_01	1
IPduGroup_03	IPduGroup_03	3
IPduGroup_04	IPduGroup_04	4
IPduGroup_05	IPduGroup_05	5
IPduGroup_06	IPduGroup_06	6

Figure 7-6 GENy setup PDU groups

If another variable name for the PDU group is needed adapt the name in the `Com` like in Figure 7-6. Pay attention to that the adapted name is the correct one. The new name is automatically updated in the CanSM see Figure 7-5.

7.1.6 Bus-off Recovery State Machine Configuration

Each bus-off recovery state machine has a bus-off-timer, which is compared with the three time parameters. These parameters are the same for each network handle, independent from the timer of the network handle. Each bus-off recovery state machine has a bus-off counter, which is compared with the two counter thresholds. These parameters don't depend on the network handle they are same for each counter. All "`Bus Off Settings`" parameters can be modified for each CAN channel separate.


GENy - [GENy3-CAN-CAN-CAN-base.gny : Channel0*]

Configurable Options	Channel0
+ General Settings	
− Miscellaneous	
− Bus Off Settings	
Bor Time Tx Ensured Ch [s]	5.0000*
Bor Counter L1 To L2 Ch	5*
Bor Time L1 Ch [s]	1.0000*
Bor Time L2 Ch [s]	3.0000*
Bor Counter L2 Err Ch	8*
Bor Disable Rx Dl Monitoring Ch	☐ *
+ IPdu Group Configuration	

Figure 7-7 GENy CAN channel bus-off configuration

Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
Bus-off recovery state machine				
`Bor Time Tx Ensured Ch [s]`	PRE-COMPILE, LINK-TIME, POST-BUILD	uint16	**5.000**	Defines the duration of the bus-off event check. If a new bus-off occurs during this time period, the CAN State Manager evaluates this bus-off as sequential bus-off without successful recovery. The configured time must be large enough to ensure that new PDUs are transmitted. **The entered value is multiplied by the Cycle Time. The result has to be covered by the 16 bit integer range are allowed. E.g. if the Cycle Time is set 0.010 only values between 0.0 and 655.3500 are allowed.**
`Bor Counter L1 To L2 Ch`	PRE-COMPILE, LINK-TIME, POST-BUILD	uint16	**5**	Defines at which bus-off-counter value the bus-off recovery state machine switches from level 1 to level 2.
`Bor Time L1 Ch [s]`	PRE-COMPILE, LINK-TIME, POST-BUILD	uint16	**1.000**	Defines the duration of the bus-off recovery time in level 1 (usually short recovery time), which is given to restart the CAN network. **The entered value is multiplied by the Cycle Time. The result has to be covered by the 16 bit integer range are allowed. E.g. if the Cycle Time is set 0.010 only values between 0.0 and 655.3500 are allowed.**
`Bor Time L2 Ch`	PRE-COMPILE,	uint16	**3.000**	Defines the duration of the bus-off

Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
`[s]`	LINK-TIME, POST-BUILD			recovery time in level 2 (usually long recovery time), which is given to restart the CAN network. **The entered value is multiplied by the Cycle Time. The result have to be covered by the 16 bit integer range are allowed. E.g. if the Cycle Time is set 0.010 only values between 0.0 and 655.3500 are allowed.**
`Bor Counter L2 Err Ch`	PRE-COMPILE, LINK-TIME, POST-BUILD	Uint16	**8**	Defines at which bus-off-counter value the bus-off recovery state machine shall report a failed production error state to the DEM. Caused by the bus-off state machine (see Figure 4-2) a counter value greater than the "CounterL1 to L2" should be chosen. The all recovery retries of level one and level two has to be performed unsuccessful before the DEM is triggered.
`Bor Disable Rx Dl Monitoring Ch`	PRE-COMPILE, LINK-TIME	define	STD_ON **STD_OFF**	The CAN State Manager allows enabling and disabling the deadline monitoring. The RX PDU group deadline monitoring may be deactivated to avoid timeouts during bus-off recovery process. If the checkbox is selected on the CanSM will enable the monitoring of the Rx PDU groups if the full communication mode is active without a bus-off.

Table 7-4 Bus-off recovery state machine parameter

Info
A value equal to zero for the “`Bor Counter L1 To L2 Ch`” is senseless because the counter is at least incremented one time see bus-off state machine in [1]

Info
A value equal or lower than “`Bor Counter L1 To L2 Ch`” for the “`Bor Counter L2 Err Ch`” is senseless because the counter is at least incremented one time after the counter reaches the threshold “`Bor Counter L1 To L2 Ch`” see bus-off state machine in [1]

7.1.7 Values from other BSW Modules

To configure the CanSM some settings in other BSW modules have to be done. The necessary configuration options in dependency to the BSW module are listed hereafter.

The DEM has to deliver for each configured CAN network one specific bus-off error event code. Values for production code event IDs are assigned externally by the configuration of the DEM. The error event codes are published in a DEM header file. The declaration and variable names, which have to follow the naming convention of the SWS [1], are generated by the configuration tool as soon as the network handles are available.

```
CANSM_E_BUSOFF_NETWORK_<X>
```

	Example The assigned bus-off error for the network handle 5 is CANSM_E_BUSOFF_NETWORK_5.The IDs of the error codes are assigned by the DEM at configuration time.

The Com defines the symbolic handles of the IPDU Groups in the configuration header of the Com. The configuration tool of the CAN State Manager maps the names of the Com to the fields of the arrays.

8 AUTOSAR Standard Compliance

8.1 Additions / Extensions

8.1.1 Additional Error Handling in the Network Mode State Machine

The NW SM checks if the setting of the controller or transceiver fails. In this case the CanSM tries to recover the failure in the next cycle.

8.1.2 API CanSM_InitMemory()

This service function was added to be called at Power On or after reset to set the global CanSM state, afterwards the CanSM can be initialized correctly.

8.1.3 Error Code

The following error codes are reported additionally to the errors defined in [1]:

> `CANSM_E_PARAM_CONTROLLER` the name is mentioned in the SWS but got no value.

> `CANSM_E_MODE_CHANGE` addition to ASR in case if transition fails several times in sequence.

> `CANSM_E_SETTRANSCEIVERMODE` addition to ASR in case if set transceiver mode fails.

8.1.4 Configuration Options

> It’s possible to deactivate the DEM at pre-compile time, like DET.

> It’s possible to change the option `Rx deadline monitoring` at link time too.

> It’s possible to enable a feature which deactivates the `Rx deadline monitoring` in the state CANSM_SILENT_COMMUNICATION too.

8.1.5 API CanSM_PreventBusSleepAtStartUp()

This service function was added to prevent the overwriting of the wakeup reason.

8.1.6 ECU passive mode

The passive mode deactivates the Tx part during full communication. The ECU listened “passive” on the CAN bus.

8.1.7 Bus-off notification

The CanSM informs the application about a bus-off.

8.1.8 Support for XCP shutdown

If XCP and CanSM is activates at the same system channel the XCP gets informed if the Tx Channel switches to Online or back again.

8.1.9 No mode notification during CanSM_Init

The ComM_BusSM_ModeIndication is not called during the transition from `CANSM_INIT` to `CANSM_NO_COMMUNNICATION` because the ComM becomes initialized after the CanSM.

8.1.10 Partial Networks

If Partial Network for a CAN channel is activated the CAN transceiver can only be woken up by a specified CAN Message. Also the network management will ignore Nm messages which don't belong to the Partial Network. If the feature is enabled (see Table 7-1 and Table 7-2) the CanSM adapts the shutdown sequence.

8.1.11 BswM ComMode Indication

If the feature is enabled (see Table 7-1) the CanSM informs the BswM about state changes via the function BswM_CanSM_CurrentState.
The transmitted states are:

- `CANSM_BSWM_NO_COMMUNICATION`
- `CANSM_BSWM_SILENT_COMMUNICATION`
- `CANSM_BSWM_FULL_COMMUNICATION`
- `CANSM_BSWM_BUS_OFF`

8.1.12 Multiple Identity

If the feature is activated the CanSM generates several configuration structures which are declared in the configuration header CanSM_Cfg.h. According to the identity the EcuM has to calls the CanSM_Init function with the according configuration.

8.1.13 Additional Dem error

The CanSM reports two additional errors to the Dem module.

> CANSM_E_MODE_CHANGE_NETWORK_<X> if the transition tries exceeds the configured number of “`Hw Repetition`“.

> CANSM_E_SETTRANSCEIVERMODE_NETWORK_<X> if the change of the transceiver mode fails.

8.1.14 Additional bus-off recovery in state silent

If a bus-off occurs outside the state FULL_COMMUNICATION, the CanSM handles the bus-off and sets the CAN controller mode back to STARTED.

8.2 Limitations

8.2.1 Controllers

The CanSM supports only one controller per channel.

9 Glossary and Abbreviations

9.1 Glossary

Term	Description
GENy	Generation tool for CANbedded and MICROSAR components
Bus communication messages	Bus communication messages are all messages sent on the communication bus. This
Bus sleep	No activity required on communication bus (e.g. CAN bus sleep).
CAN communication matrix	Describes the complete CAN network: Participating nodes Definition of all CAN PDUs (identifier, DLC) Source and Sinks for PDUs
CAN controller	A CAN controller is a CPU on-chip or external standalone hardware device. One CAN controller is connected to one physical channel.
CAN device driver	Generic term of CAN Driver and CAN Transceiver Driver.
CAN L-PDU	CAN Protocol Data Unit. Consists of an identifier, DLC and data (SDU).
CAN L-SDU	CAN Service Data Unit. Data that are transported inside the CAN L-PDU.
Diagnostic PDU scheduling	Sending of diagnostic PDUs. Scheduling of CAN diagnostic PDUs is performed by the diagnostic module, scheduling of LIN diagnostic PDUs is performed by the diagnostic module and the LIN interface and scheduling of FlexRay diagnostic PDUs is performed
L-PDU channel group	Group of CAN L-PDUs, which belong to just one underlying network. Usually they are handled by one upper layer module.
L-PDU handle	The L-PDU handle is defined as integer type and placed inside the CAN Interface layer. Typically each handle represents an L-PDU, which is a constant structure with information for Tx/Rx processing.
Passive wake-up	Wakeup by another ECU and propagated (e.g. by bus or wakeup-line) to the ECU currently in focus.

Table 9-1 Glossary

9.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
ASR	AUTOSAR
AUTOSAR	Automotive Open System Architecture
BOR SM	Bus Of Recovery State Machine
BSW	Basis Software
CAN	Controller Area Network
CanIf	CAN Interface
CanSM	CAN State Manager
Cbk	call-back / call-out notification (functions)
Cfg	Configuration

Com	Communication (means the ASR module)
ComM	Communication Manager
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DTC	Diagnostic Trouble Code
ECU	Electronic Control Unit
EcuM	ECU State Manager
EMI	electromagnetic interference
GUI	Graphical User Interface
HIS	Hersteller Initiative Software
ISR	Interrupt Service Routine
MemMap	Memory Mapping
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
NW SM	NetWork State Machine
PB	Post Build
PC	Personal Computer
PDU	Protocol Data Unit
RTE	Runtime Environment
SchM	BSW Scheduler
SRS	Software Requirement Specification
SWC	Software Component
SWS	Soft Ware Specification (here it usually means the SWS_CAN_StateManager)
XCP	X Calibration Protocol (X means able to handle several bus systems CAN, FlexRay)

Table 9-2 Abbreviations

10 Contact

Visit our website for more information on

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

www.vector-informatik.com