MICROSAR FlexRay State Manager

Technical Reference

Version 1.2.0

Authors	Mark A. Fingerle
Status	Released

Document Information

History

Author	Date	Version	Remarks
Mark A. Fingerle	2012-08-08	1.0.0	Creation from scratch
Mark A. Fingerle	2014-10-13	1.1.0	ESCAN00076761 Post-Build Selectable (Identity Manager) support 6.1.3 ESCAN00075457 Add support for delayed FlexRay communication cluster shutdown 3.3.8, 6.2.5 ESCAN00079339 Description BCD-coded return-value of GetVersionInfo()
Mark A. Fingerle	2016-05-13	1.2.0	Add missing API 5.2.8 FrSM_SetEcuPassive FEAT-2724 Handle several FlexRay clusters Table 3-1 Supported AUTOSAR standard conform features, Table 3-2 Not supported AUTOSAR standard conform features

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	Specification of FlexRay State Manager	2.2.0
[2]	AUTOSAR	Specification of Development Error Tracer	3.2.0
[3]	AUTOSAR	Specification of Diagnostics Event Manager	4.2.0
[4]	AUTOSAR	List of Basic Software Modules	1.6.0
[5]	AUTOSAR	Specification of FlexRay Interface	3.3.0
[6]	AUTOSAR	Specification of Communication Manager	4.0.0
[7]	AUTOSAR	Specification of Basic Software Mode Manager	1.2.0

Scope of the Document

This technical reference describes the general use of the FlexRay State Manager basis software. All aspects which are FlexRay controller specific are described in the technical reference of the FlexRay Interface, which is also part of the delivery.

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
1.0.0	Creation according to AUTOSAR 4.0.3

Table 1-1 Component History

2 Introduction

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

Supported AUTOSAR Release*:	4	
Supported Configuration Variants:	pre-compile, Post-Build Selectable	
Vendor ID:	FRSM_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	FRSM_MODULE_ID	142 decimal (according to ref. [4])

* For the precise AUTOSAR Release 4.x please see the release specific documentation.

The FlexRay State Manager (FrSM) realizes a software layer between the Communication Manager (ComM) and the FlexRay Interface (FrIf). The FrSM handles the startup and shutdown of the communication of a FlexRay cluster. The FrSM maps the FrIf states to the states of the ComM and causes the necessary actions to change the FrIf state to these requested by the ComM. The main function of the FrSM is cyclically called by the Schedule Manager (SchM).

2.1 Architecture Overview

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

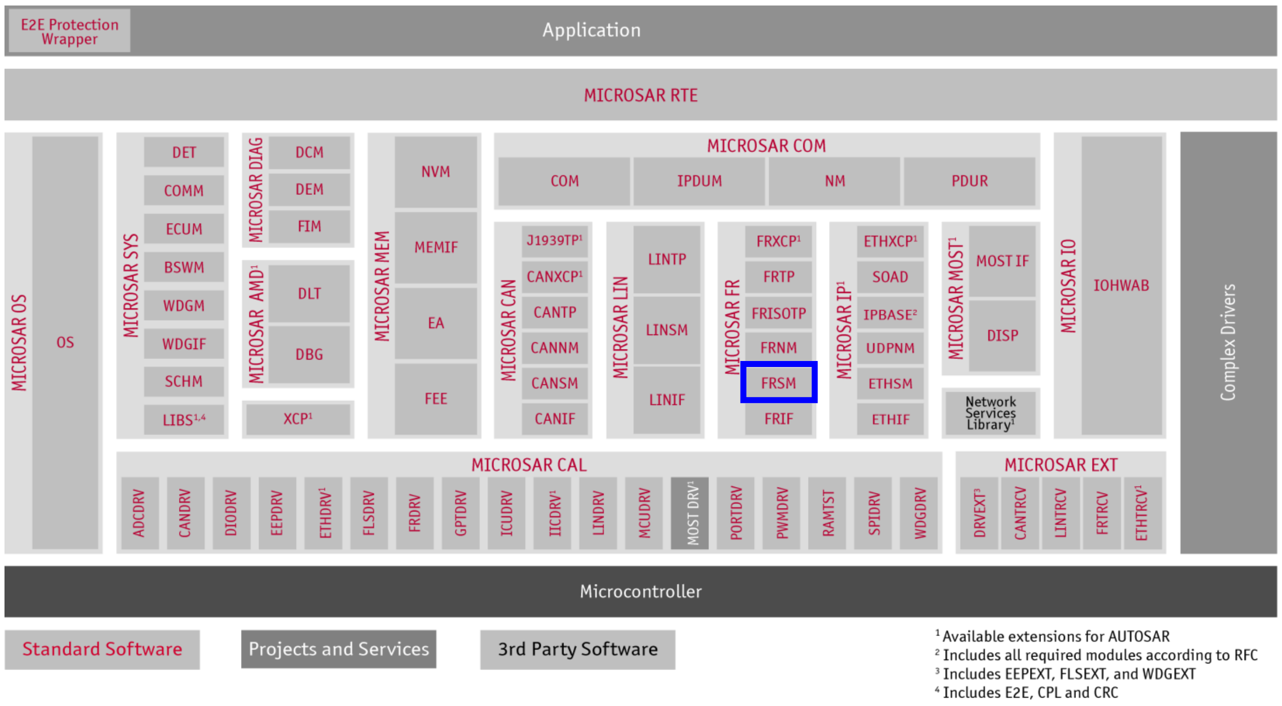

Figure 2-1 AUTOSAR architecture

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

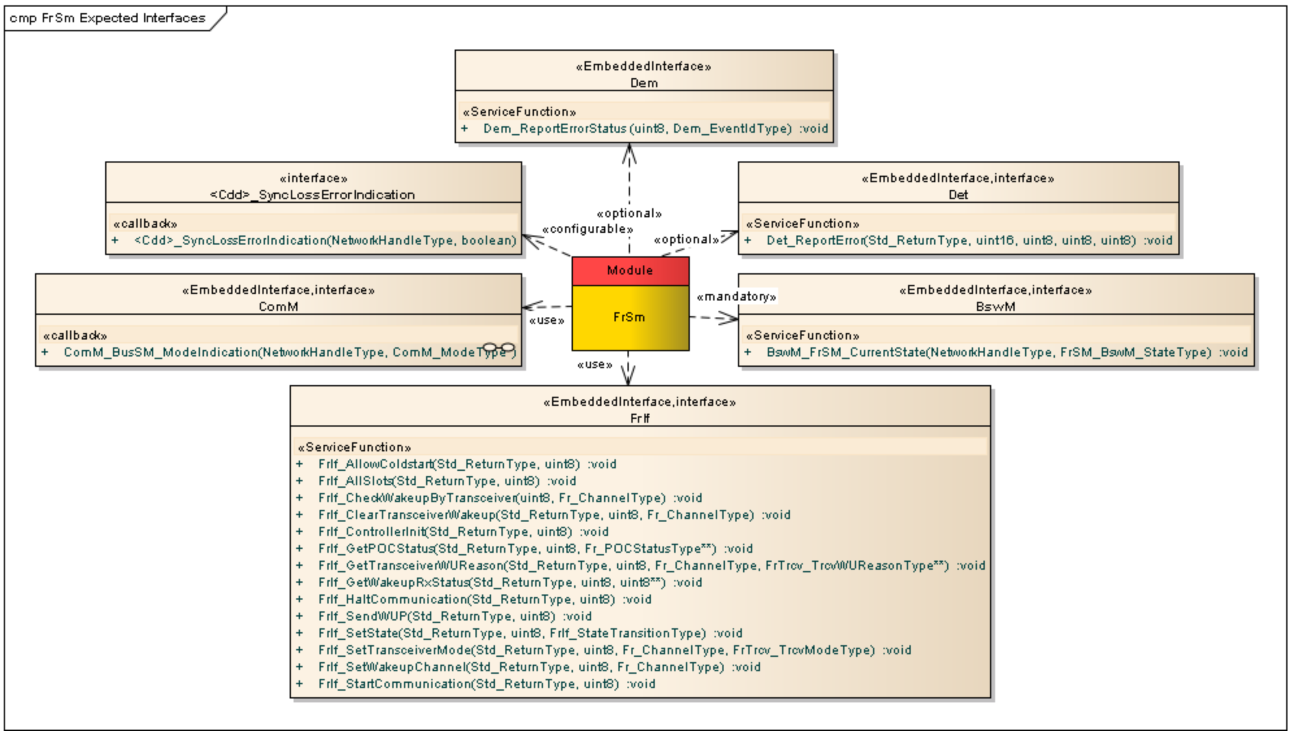
cmp FrSm Expected Interfaces
«EmbeddedInterface»
Dem
«ServiceFunction»
+ Dem_ReportErrorStatus (uint8, Dem_EventIdType) :void
«interface»
<Cdd>_SyncLossErrorIndication
«callback»
+ <Cdd>_SyncLossErrorIndication(NetworkHandleType, boolean)
«EmbeddedInterface,interface»
Det
«ServiceFunction»
+ Det_ReportError(Std_ReturnType, uint16, uint8, uint8, uint8) :void
«optional»
«configurable»
«optional»
Module
FrSm
«mandatory»
«use»
«use»
«EmbeddedInterface,interface»
ComM
«callback»
+ ComM_BusSM_ModeIndication(NetworkHandleType, ComM_ModeType)
«EmbeddedInterface,interface»
BswM
«ServiceFunction»
+ BswM_FrSM_CurrentState(NetworkHandleType, FrSM_BswM_StateType) :void
«EmbeddedInterface,interface»
FrIf
«ServiceFunction»
+ FrIf_AllowColdstart(Std_ReturnType, uint8) :void
+ FrIf_AllSlots(Std_ReturnType, uint8) :void
+ FrIf_CheckWakeupByTransceiver(uint8, Fr_ChannelType) :void
+ FrIf_ClearTransceiverWakeup(Std_ReturnType, uint8, Fr_ChannelType) :void
+ FrIf_ControllerInit(Std_ReturnType, uint8) :void
+ FrIf_GetPOCStatus(Std_ReturnType, uint8, Fr_POCStatusType**) :void
+ FrIf_GetTransceiverWUReason(Std_ReturnType, uint8, Fr_ChannelType, FrTrcv_TrcvWUReasonType**) :void
+ FrIf_GetWakeupRxStatus(Std_ReturnType, uint8, uint8**) :void
+ FrIf_HaltCommunication(Std_ReturnType, uint8) :void
+ FrIf_SendWUP(Std_ReturnType, uint8) :void
+ FrIf_SetState(Std_ReturnType, uint8, FrIf_StateTransitionType) :void
+ FrIf_SetTransceiverMode(Std_ReturnType, uint8, Fr_ChannelType, FrTrcv_TrcvModeType) :void
+ FrIf_SetWakeupChannel(Std_ReturnType, uint8, Fr_ChannelType) :void
+ FrIf_StartCommunication(Std_ReturnType, uint8) :void

Figure 2-2 Interfaces to adjacent modules of the FrSM

Applications do not access the services of the BSW modules directly.

3 Functional Description

3.1 Features

The features listed in the following tables cover the complete functionality specified for the FrSM.

The AUTOSAR standard functionality is specified in [1], the corresponding features are listed in the tables

> Table 3-1 Supported AUTOSAR standard conform features

> Table 3-2 Not supported AUTOSAR standard conform features

For further information of not supported features see also chapter 6.

Vector Informatik provides further FrSM functionality beyond the AUTOSAR standard. The corresponding features are listed in the table

> Table 3-3 Features provided beyond the AUTOSAR standard

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
Translation of network communication mode requests
Output of current network communication modes (Polling and Callback)
Control of peripherals (Fr Transceivers, Fr Controllers)
Handle the Network mode via a separate state machine per network
Error classification, detection and notification
Handle several FlexRay clusters

Table 3-1 Supported AUTOSAR standard conform features

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

Category	Description	ASR Version
Functional	Several controllers per cluster.	4.0.3
Functional	Low number of coldstarters neither detection nor handling	4.0.3
Config	Change networks and controllers via Post-build configuration.	4.0.3
Config	Configuration variant “link-time”.	4.0.3

Table 3-2 Not supported AUTOSAR standard conform features

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond The AUTOSAR Standard
It’s possible to deactivate the Dem at pre-compile time, like Det.
MICROSAR Identity Manager using Post-Build Selectable

Table 3-3 Features provided beyond the AUTOSAR standard

3.2 Initialization

Some variables need a default value even before the initialization function will be called. The initialization of these variables is usually done by the startup code of the ECU. Otherwise the `FrSM_InitMemory` function has to be called before the initialization function. Then the initialization function of the FrSM has to be called. The FrSM gets the configuration data of the current system like the number of available clusters and an own configuration record per cluster. If the development error detection is activated the FrSM handles several validity checks at the configuration data. Then the internal variables of the state machine are set to their start values.

3.3 State Machine

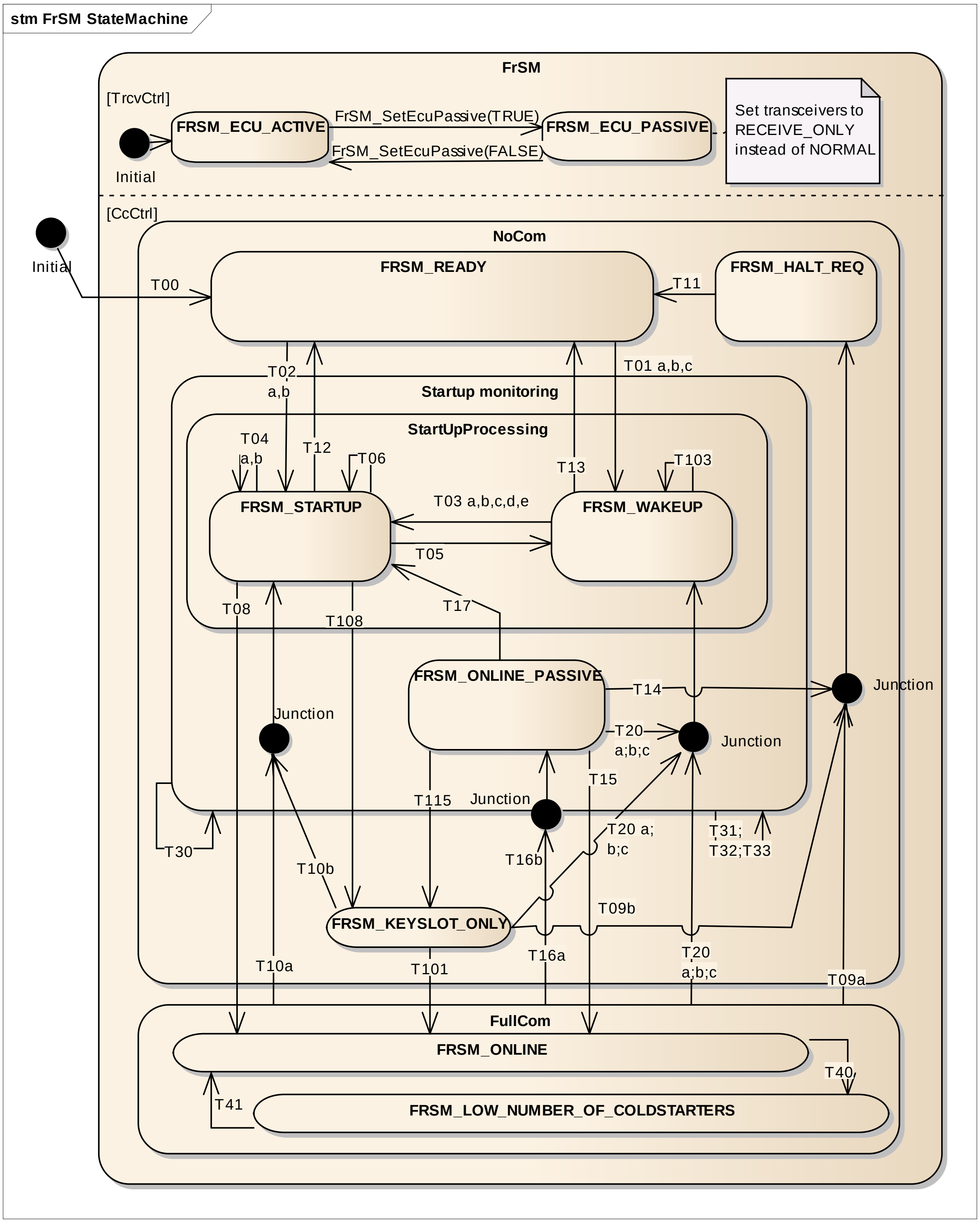

Figure 3-1 State machine of the FrSM

3.3.1 Meta States FullCom / NoCom

In the FullCom state the network is started and communication works well. In the NoCom state the network isn't started or the communication controller may have detected some errors. The ComM is informed at each time the Meta state changes.

3.3.2 FRSM_INITIAL

When the ECU is powered on the initialization function of the FrSM is called. The initialization function sets the variables to the default value, maps the configuration data to the local handle and Initializes the FlexRay controllers for all clusters of the configuration. The state of the FrSM is now `FRSM_READY`. After initialization the API of the FrSM are ready for use.

3.3.3 FRSM_STARTUP / FRSM_WAKEUP / FRSM_READY

If the FrSM receives a full communication request it sets the transceiver in `normal mode` (T01 or T02) and starts the FlexRay communication (T03 or T02). Before the FrSM triggers the transition T01 it detects the wake-up reason. In case the ECU was woken up by bus (wake-up has been detected for all of the FlexRay channels of the cluster to which the ECU is connected) the bus is already awake and the execution of the channel wake-up isn't needed. In that case the FrSM executes the transition T02 instead T01 even the ECU is configured as wake-up node. Depending on the configuration a channel wake-up (T01) and a cold start process (T02 or T04) is initiated too. The transition ends in state `FRSM_STARTUP`. Now the startup of the controller should be succeeded and the FrSM sets the FlexRay Interface to `ONLINE`, triggers a Dem status passed and updates the internal state to `FRSM_ONLINE` (T08).

If the ECU is connected to both FlexRay channels of the cluster and T01 becomes executed (see above) the function `FrIf_SetWakeupChannel` is always called before `FrIf_SendWUP`. In case no wake-up has been detected the wake-up channel is set to CHANNEL_A otherwise to the channel on which no wake-up has been detected.

If the configuration parameter `FrSMDelayStartupWithoutWakeup` is activated the "allow cold start" in transition T02 is skipped and the timer t1 is started. So the first "allow cold start" attempt is delayed by "Duration T1" until the transition T04 is executed. If "Duration T1" is configured to zero there will be also no difference to the "normal" behavior.

If the communication controller has not reached `normal active` during the time `Duration T2` the wake-up process is repeated. The wake-up might be repeated as often as specified in `FrSMStartupRepetitionsWithWakeup`, of course only if the CC state isn't "`Normal Active`" and still full communication is requested. The wake-up process includes the initialization of the controller, the channel wake-up (T05), the start of the FlexRay communication (T03) and if configured a cold start (T04). The wake-up repetition also depends on the wake-up reason. In case the ECU was woken up by bus (passive wake-up) the bus is already awake and the execution of the channel wake-up isn't needed. In that case the FrSM acts like the option `FrSMIsWakeupEcu` is deactivated and executes the transition T06 instead T05.

If all specified `FrSMStartupRepetitionsWithWakeup` have been executed, the startup repetitions without a wake-up follow. The number `FrSMStartupRepetitions` includes all repetitions as well the startup with and without a wake-up.

The startup repetitions without a wake-up (T06) might be repeated as many times as specified in the `FrSMStartupRepetitions` less the `FrSMStartupRepetitionsWithWakeup` (which has been already performed). Of course the transition T06 will be skipped, if the CC state is `NORMAL_ACTIVE` or `COMM_NO_COMMUNICATION` is requested. If all specified `Startup Repetitions` have been executed and the controller is still not active the FrSM stays in the state `FRSM_STARTUP` and stops any startup retry.

It's also possible to repeat the startups infinitely. This happens if no threshold of the corresponding value has been configured or the corresponding checkbox in the configuration tool has been selected. If the number of repetitions with wake-up is set to infinite the value of the repetitions without wake-up are of course irrelevant.

Caution
Unlimited number of startup attempts could lead to a continuous disturbance of the FlexRay communication.

The ComM may stop the startup by requesting no communication. In this case the FrSM sets the transceiver into standby, enables the wake-up, initializes the controller and sets the internal state to `FRSM_READY` (T12 or T13). These two transitions are executed immediately.

3.3.4 FRSM_WAKEUP, Send Multiple Wake-Up Pattern

The FrSM may stay in the state `WAKEUP` and send a configurable number of wake-up patterns. The intention is to send additional WUP during the gap between the first WUP (T01) and the cold start (after T04).
If the "`FrSMNumWakeupPatterns`" is set to one (WUP in T01) there is no difference to the behavior described above. Otherwise the FrSM will send wake-up patterns, until the configured "`FrSMNumWakeupPatterns`" are reached (including T01 or T05) or external bus communication is detected. In this case the FrSM will perform the transition T03 to `STARTUP`. If any of the WUP (of this startup repetition) has been transmitted successfully (`POC.WakeupStatus == WAKEUP_TRANSMITTED`) or the time which has been elapsed (in state `WAKEUP`) exceeds the "Duration T1" or if a dual channel system has been woken up on exact one channel (`PARTIAL_WU`) then `ALLOW_COLDSTART` will be triggered immediately.

3.3.5 FRSM_STATE_ONLINE

In this state the network is started and communication works well. The FrSM waits till it receives a No Communication Request. In this case the FrSM stops the PDU groups, sets the FlexRay Interface to `OFFLINE`, stops the FlexRay communication and sets the internal state to `FRSM_HALT_REQ` (T09). This transition is executed immediately.

Under certain circumstances some failures on the bus triggers the CC to switch the state. If the state of the CC has switched to halt, freeze or ready the FrSM stops the PDU groups, sets the FlexRay Interface to `OFFLINE`, initializes the controller, starts the FlexRay communication and sets the internal state to `FRSM_STARTUP` (T10) if `FrSMCheckWakeupReason` has been selected, otherwise to `FRSM_WAKEUP` (T20). If the state of the CC has switched to `normal passive` the FrSM stops the PDU groups and sets the internal state to `FRSM_ONLINE_PASSIVE` (T16).

3.3.6 FRSM_KEY_SLOT_ONLY

Before the state `FRSM_ONLINE` will be entered in case the POCstate is `normal active` the FrSM check the SlotMode of the FlexRay controller. If the SlotMode is not equal to `FR_SLOTMODE_ALL` the FrSM enters the state `FRSM_KEYSLOT_ONLY` instead of `FRSM_ONLINE`. To leave the KeySlotOnlyMode the function `FrSM_AllSlots()` has to be called by the BswM. Within `FrSM_AllSlots()` the FrSM calls the similar named function `FrIf_AllSlots()` of the FrIf where the SlotMode switches.

Apart from that the FrSM reacts on communication requests or changes of the POC state in the same way as in FRSM_STATE_ONLINE.

3.3.7 FRSM_STATE_ONLINE_PASSIVE

In this state some disturbance on the bus has been occurred and the CC has left the `normal active` state.

The FrSM waits till it receives a no communication request. In this case the FrSM sets the FlexRay Interface to `OFFLINE`, stops the FlexRay communication and sets the internal state to `FRSM_HALT_REQ` (T14). This transition is executed immediately. If the CC has already switched to the `halt` state the FrSM sets the FlexRay Interface to `OFFLINE`, initializes the controller, starts the FlexRay communication and sets the internal state to `FRSM_STARTUP` (T17). If the CC has already switched to the `normal active` state the FrSM stops the PDU groups, and updates the internal state to `FRSM_ONLINE` (T15) or to `FRSM_KEY_SLOT_ONLY` (T115).

3.3.8 FRSM_STATE_HALT_REQ

The FrSM waits till the communication controller reaches the halt state. In this case the FrSM executes the transition T11 sets the transceiver in standby, enables the wake-up, initializes the controller and sets the internal state to FRSM_READY.

In some cases the instant deactivation of the transceiver might cause an unintentional wakeup. To avoid these, the transition T11 from FRSM_HALT_REQ to FRSM_READY may be delayed by the configured “Trcv Stby Delay Timer” value. The timer is handled as zero if no transceiver is configured. The timer runs only if the POC state of the FlexRay controller is in the HALT mode. The delay timer will be ignored if the FrSM gets a new FullCom request.

3.3.9 Startup Monitoring (FRSM_STARTUP / FRSM_WAKEUP / FRSM_STATE_HALT_REQ)

The timer t3 is used to supervise the startup/restart process. The timer t3 runs if the FrSM is in the state group `FRSM_STATE_ONLINE_PASSIVE`, `FRSM_WAKEUP` and `FRSM_STARTUP`. The timer t3 is started if this state group is entered.

If the timer expires, a Dem failed message is triggered once. If the timer has been expired the `SyncLossErrorIndication` function gets called every `FrSM_MainFunction` until the timer becomes deactivated. The timer t3 is deactivated if this state group `FRSM_STARTUP`, `FRSM_WAKEUP` and `FRSM_STATE_ONLINE_PASSIVE` is left.

The function name of the `SyncLossErrorIndication` function can be configured in the configuration tool via the parameter `FrSMSyncLossErrorIndicationName`. Usually the function `FrNm_StartupError` of the FrNm is chosen.

3.4 Main Functions

There is one main function for each cluster available. If the development error detection is activated the function checks if the cluster index is valid and if the FrSM has been initialized. Then the function detects if the conditions of a transition are fulfilled, triggers the necessary actions by calling API functions of other BSW and adapts the internal data.

3.4.1 Communication Modes

The ComM collects the communication requests from the SWC and from the network. Accordingly the ComM calculates the needed communication mode and requests this from the FlexRay State Manager via the function `FrSM_RequestComMode`. The function `FrSM_RequestComMode` checks the function parameter, stores the requested communication mode and in some cases handles transitions directly to stop the transmission immediately.

3.4.2 Communication Mode Polling

The ComM is informed about the changes of the Meta states NoCom and FullCom via the callback function `ComM_BusSM_ModeIndication`.

Additionally the ComM has the possibility to request the current communication mode via the API function `FrSM_GetCurrentComMode`. The function delivers the communication mode of a specific cluster so it needs the network handle and an address reference where the mode shall be stored to.

3.5 Error Handling

3.5.1 Development Error Reporting

By default, development errors are reported to the Det using the service `Det_ReportError()` as specified in [2], if development error reporting is enabled (i.e. pre-compile parameter `FRSM_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 FrSM ID is 142.

The reported service IDs identify the services which are described in 5.2. The following table presents the service IDs and the related services:

Service ID	Service
0x01	`FrSM_Init`
0x80 + `Cluster_Id`	`FrSM_MainFunction_<ClusterId>`
0x02	`FrSM_RequestComMode`
0x03	`FrSM_GetCurrentComMode`

Service ID	Service
0x04	`FrSM_GetVersionInfo`
0x05	`FrSM_AllSlots`
0x06	`FrSM_SetEcuPassive`

Table 3-4 Service IDs

Note
The Service ID 0x03 used twice (FrSM_RequestComMode and FrSM_GetCurrentComMode). So 0x02 is used for FrSM_RequestComMode like in the prior AUTOSAR 3.x releases.

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

Error Code		Description
`0x01`	`FRSM_E_NULL_PTR`	Invalid pointer in parameter list
`0x02`	`FRSM_E_INV_HANDLE`	Invalid network handle parameter
`0x03`	`FRSM_E_UNINIT`	FrSM module was not initialized
`0x04`	`FRSM_E_INV_MODE`	Invalid communication mode requested

Table 3-5 Errors reported to Det

3.5.2 Production Code Error Reporting

By default, production code related errors are reported to the Dem using the service `Dem_ReportErrorStatus()` as specified in [3], if production error reporting is enabled (i.e. pre-compile parameter `FRSM_PROD_ERROR_DetECT==STD_ON`).

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

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

Error Code	Description
`FRSM_E_CLUSTER_STARTUP`	Value to identify the FrSM to the Dem. A failed message is sent if the FlexRay could not reach the state normal active within the configured time or a passed message if the startup worked.

`FRSM_E_CLUSTER_SYNC_LOSS`	Value to identify the FrSM to the Dem. A failed message is sent if the FrSM is in state ONLINE and the POC state of the FlexRay controller has been changed automatically (NORMAL_PASSIVE, HALT or FREEZE) (T10, T16). The corresponding passed message is sent in the transitions. T08, T12, T13, T14 and T15.

Table 3-6 Errors reported to Dem

4 Integration

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

4.1 Scope of Delivery

The delivery of the FrSM contains the files which are described in the chapters 4.1.1 and 4.1.2:

4.1.1 Static Files

File Name	Source Code Delivery	Object Code Delivery	Description
FrSM.c	■		This is the source file of the FrSM, which contains all API functions and algorithms of the FrSM. (not available if libraries are delivered)
FrSM.h	■	■	This is the header file of the FrSM, which contains the API prototypes and definitions.
FrSM_Types.h	■	■	This header file contains the global data types of the FrSM.

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool [config tool].

File Name	Description
FrSM_Cfg.h	This is the configuration header file of the FrSM, which contains the configurable parameters. The setup has to be made before compiling the FrSM sources.
FrSM_Lcfg.c	This is the link time configuration file of the FrSM. The type declarations which are made in the FrSM_Types.h header can be defined with the currently needed values. The file may be implemented, compiled after the FrSM sources and linked to the object code.
FrSM_PBcfg.c	This is the post-build time configuration file of the FrSM. The type declarations which are made in the FrSM_Types.h header may be defined with the currently needed values. The file is handled in the same way as the link time configuration file. Additionally the memory addresses of the values are known and fix. So the values can be replaced after the build process too.

Table 4-2 Generated files

4.2 Include Structure

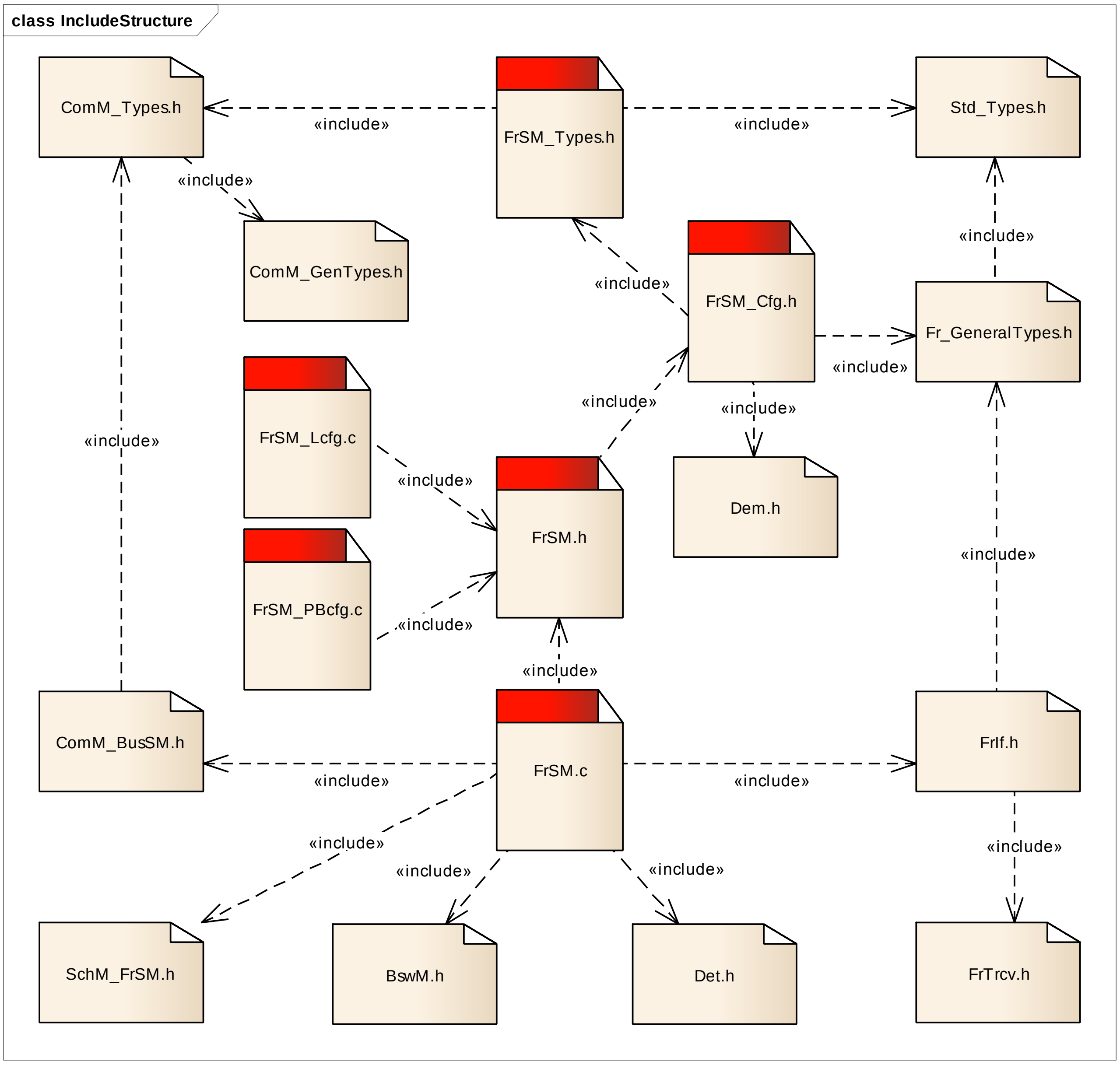

Figure 4-1 Include structure

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 FrSM and illustrates their assignment among each other.

Compiler Abstraction Definitions / Memory Mapping Sections	FRSM_VAR_NOINIT	FRSM_VAR_ZERO_INIT	FRSM_CONST	FRSM_PBCFG	FRSM_CODE	FRSM_APPL_VAR	FRSM_APPL_CODE
FRSM_START_SEC_PBCFG_ROOT FRSM_STOP_SEC_PBCFG_ROOT				■			
FRSM_START_SEC_PBCFG FRSM_STOP_SEC_PBCFG				■			
FRSM_START_SEC_CONST_8BIT FRSM_STOP_SEC_CONST_8BIT			■				
FRSM_START_SEC_CONST_32BIT FRSM_STOP_SEC_CONST_32BIT			■				
FRSM_START_SEC_CONST_UNSPECIFIED FRSM_STOP_SEC_CONST_UNSPECIFIED			■				
FRSM_START_SEC_VAR_ZERO_INIT_8BIT FRSM_STOP_SEC_VAR_ZERO_INIT_8BIT		■					
FRSM_START_SEC_VAR_NOINIT_8BIT FRSM_STOP_SEC_VAR_NOINIT_8BIT	■						
FRSM_START_SEC_VAR_NOINIT_16BIT FRSM_STOP_SEC_VAR_NOINIT_16BIT	■						
FRSM_START_SEC_VAR_NOINIT_UNSPECIFIED FRSM_STOP_SEC_VAR_NOINIT_UNSPECIFIED	■						
FRSM_START_SEC_CODE FRSM_STOP_SEC_CODE					■	■	■

Table 4-3 Compiler abstraction and memory mapping

4.4 Critical Sections

Critical sections are handled by the BSW Scheduler. To ensure data consistency and a correct function of the FrSM the following exclusive areas have to be provided during the integration. The chosen critical section solution has to be configured that it’s ensured that the API functions do not interrupt each other.

> The `FRSM_EXCLUSIVE_AREA_0` has to be used if it is possible that the function `FrSM_MainFunction_<Idx>()` is interrupted by the function `FrSM_RequestComMode()`. It is recommended to use AUTOSAR OS ‘Resources’ for these exclusive areas to prevent priority inversions and dead-locks.

> The `FRSM_EXCLUSIVE_AREA_1` has to be used if it is possible that the function `FrSM_MainFunction_<Idx>()` is interrupted by the function `FrSM_SetEcuPassive()`.

- The `FRSM_EXCLUSIVE_AREA_2` has to be used if it is possible that the function `FrSM_RequestComMode()` is interrupted by any of the functions `FrSM_SetEcuPassive()` or `FrSM_MainFunction_<Idx>()`.
- The `FRSM_EXCLUSIVE_AREA_3` has to be used if it is possible that the function `FrSM_SetEcuPassive()` is interrupted by the function `FrSM_RequestComMode()`.

5 API Description

For an interfaces overview please see Figure 2-2.

5.1 Type Definitions

The types defined by the FrSM are described in this chapter.

Type Name	C-Type	Description	Value Range
`FrSM_BswM_StateType`	`uint8`	This type defines the states of the FrSM state machine which are indicated to the BswM.	`FRSM_INITIAL` FrSM is not initialized in this state
			`FRSM_BSWM_READY` Mapped FlexRay CC state POC:ready or (transitional) POC:default config or (transitional) POC:config or (transitional) POC:halt
			`FRSM_BSWM_STARTUP` Mapped FlexRay CC state POC:start-up
			`FRSM_BSWM_WAKEUP` Mapped FlexRay CC state POC:wake-up
			`FRSM_BSWM_HALT_REQ` Mapped FlexRay CC state POC:normal active or POC:normal passive
			`FRSM_BSWM_KEYSLOT_ONLY` Mapped FlexRay CC state POC:normal active but the SlotMode is not `FR_SLOTMODE_ALL`
			`FRSM_BSWM_ONLINE` Mapped FlexRay CC state POC:normal active Part of the Meta state `FullCom` Requested by ComM `COMM_FULL_COMMUNICATION`
			`FRSM_BSWM_ONLINE_PASSIVE` Mapped FlexRay CC state POC:normal passive
			`NoCom` Meta state contains each state except `FRSM_BSWM_ONLINE.` Requested by ComM `COMM_NO_COMMUNICATION`

Type Name	C-Type	Description	Value Range
`FrSM_ConfigType`	`struct`	Structure which contains the global configuration data.	
`FrSMSyncLossErrorIndicationFctPtrType`	pointer	Specifies the function that shall be called on “loss of synchronization”.	

Table 5-1 Type definitions

5.2 Services Provided by FrSM

5.2.1 FrSM_InitMemory

Prototype	
`void **FrSM_InitMemory** (void)`	
Parameter	
void	none
Return code	
void	none
Functional Description	
Function for *_INIT_*-variable initialization.	
Particularities and Limitations	
Module must not be initialized Service to initialize module global variables at power up. This function can be used to initialize the Variables in *_INIT_* sections in case they are not initialized by the startup code.	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-2 FrSM_InitMemory

5.2.2 FrSM_Init

Prototype	
`void **FrSM_Init** (const FrSM_ConfigType *const ConfigPtr)`	
Parameter	
ConfigPtr [in]	Configuration structure for initializing the module
Return code	
void	none

Functional Description
Initialization function.
Particularities and Limitations
> Interrupts are disabled. FrSM_InitMemory has been called unless FRSM_Initialized is initialized by start-up code. This function initializes the module. It initializes all variables and sets the module state to READY.
Call context
> TASK > This function is Synchronous > This function is Non-Reentrant

Table 5-3 FrSM_Init

5.2.3 FrSM_MainFunction_<Cluster Id>

Prototype	
`void FrSM_MainFunction_<Cluster Id>( void )`	
Parameter	
None	-
Return code	
None	-
Functional Description	
The main function of the FrSM triggers the state functions according to the current state. The function must be called cyclically with the configured cycle time.	
Particularities and Limitations	
■ Service ID: see table 'Service IDs' ■ FrSM has to be initialized. Function has to be called cyclically. The cycle time is set in the configuration tool. ■ Reentrant for different FlexRay cluster ID suffix	
Expected Caller Context	
■ Cyclically on task level	

Table 5-4 FrSM_MainFunction_<Cluster Id>

5.2.4 FrSM_RequestComMode

Prototype	
`Std_ReturnType FrSM_RequestComMode (NetworkHandleType NetworkHandle, ComM_ModeType ComM_Mode)`	
Parameter	
NetworkHandle [in]	FlexRay cluster for which a communication mode is requested
ComM_Mode [in]	Requested communication mode (Range: COMM_NO_COMMUNICATION, COMM_FULL_COMMUNICATION)

Return code	
Std_ReturnType	E_NOT_OK - function has been called with invalid parameters E_OK - request accepted
Functional Description	
Initiates the sequence to reach the requested communication mode.	
Particularities and Limitations	
This API function is used by the ComM to startup or shutdown the communication on a FlexRay cluster.	
Call context	
> ANY > This function is Reentrant	

Table 5-5 FrSM_RequestComMode

5.2.5 FrSM_GetCurrentComMode

Prototype	
`Std_ReturnType` **`FrSM_GetCurrentComMode`** `(NetworkHandleType NetworkHandle, ComM_ModeType *ComM_ModePtr)`	
Parameter	
NetworkHandle [in]	FlexRay cluster for which a communication mode is requested
ComM_ModePtr [out]	Pointer to the memory location where the current communication mode shall be stored
Return code	
Std_ReturnType	E_OK Request was accepted E_NOT_OK Request was not accepted
Functional Description	
Reports the communication mode.	
Particularities and Limitations	
-Reports the last communication mode which has been passed to the ComM	
Call context	
> ANY > This function is Synchronous > This function is Reentrant	

Table 5-6 FrSM_GetCurrentComMode

5.2.6 FrSM_GetVersionInfo

Prototype	
`void` **`FrSM_GetVersionInfo`** `(Std_VersionInfoType *versioninfo)`	
Parameter	
versioninfo [out]	Pointer to version information. Parameter must not be NULL (BSW00407).

Return code	
void	none
Functional Description	
Returns the version information.	
Particularities and Limitations	
-Returns the software version, vendor ID and AUTOSAR module ID of the component. Configuration Variant(s): FRSM_VERSION_INFO_API	
Call context	
> ANY > This function is Synchronous > This function is Reentrant	

Table 5-7 FrSM_GetVersionInfo

5.2.7 FrSM_AllSlots

Prototype	
`Std_ReturnType` **`FrSM_AllSlots`** `(NetworkHandleType NetworkHandle)`	
Parameter	
NetworkHandle [in]	FlexRay cluster for which a communication mode is requested
Return code	
Std_ReturnType	E_OK Request was successful E_NOT_OK Request was not successful, any error occurred
Functional Description	
Calls the all slot function.	
Particularities and Limitations	
FrSM initialization Forwarding the function call to the FrIf Configuration Variant(s): FRSM_ALLSLOTS_SUPPORT -	
Call context	
> ANY > This function is Synchronous > This function is Reentrant	

Table 5-8 FrSM_AllSlots

5.2.8 FrSM_SetEcuPassive

Prototype	
`Std_ReturnType` **`FrSM_SetEcuPassive`** `(boolean FrSM_Passive)`	
Parameter	
FrSM_Passive [in]	parameter is true if Ecu Mode should be passive

Return code	
Std_ReturnType	E_OK Request was successful E_NOT_OK Request was not successful, any error occurred
Functional Description	
This API function notifies the FrSM if ECU has to be set in passive mode.	
Particularities and Limitations	
FrSM initialization Transceiver is set to mode receive only if passive mode is activated. Each cluster needs transceivers to set the whole ECU in passive mode. Otherwise the clusters without transceivers may still be able to transmit messages. Configuration Variant(s): FRSM_ECU_PASSIVE_MODE	
Call context	
> ANY > This function is Synchronous > This function is Non-Reentrant	

Table 5-9 FrSM_SetEcuPassive

5.3 Services Used by FrSM

In the following table services provided by other components, which are used by the FrSM are listed. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
BswM	BswM_FrSM_CurrentState
FrIf	FrIf_ControllerInit
FrIf	FrIf_StartCommunication
FrIf	FrIf_HaltCommunication
FrIf	FrIf_SetWakeupChannel
FrIf	FrIf_SendWUP
FrIf	FrIf_SetState
FrIf	FrIf_SetTransceiverMode
FrIf	FrIf_GetWakeupRxStatus
FrIf	FrIf_GetTransceiverWUReason
FrIf	FrIf_ClearTransceiverWakeup
FrIf	FrIf_GetPOCStatus
FrIf	FrIf_AllowColdstart
FrIf	FrIf_AllSlots
FrIf	FrIf_GetState
<Cdd>	<Cdd>_SyncLossErrorIndication
ComM	ComM_BusSM_ModeIndication
Dem	Dem_ReportErrorStatus

Component	API
Det	Det_ReportError

Table 5-10 Services used by the FrSM

6 AUTOSAR Standard Compliance

6.1 Additions/ Extensions

6.1.1 API FrSM_InitMemory()

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

6.1.2 Configuration Options

It's possible to (de)activate the Dem at pre-compile time, like Det.

6.1.3 Post-Build Selectable (Identity Manager)

The code generator and the static code supports Post-Build Selectable configuration.

6.2 Limitations

6.2.1 Controllers

The FrSM supports only one controller per cluster.

6.2.2 Shy of Coldstarter

The detection of low number of coldstarters is not supported.

6.2.3 Configuration Class

Only VARIANT-PRE-COMPILE and Post-Build Selectable are supported.

6.2.4 No Dual Channel Wake-up Echo

In case of a PARTIAL_WU_BY_BUS the WUP is not echoed to the waking channel. Instead of the echo the wake-up channel forward is performed. If multiple WUPs on the wake-up channel are needed the `FrSMNumWakeupPatterns` can be increased.

6.2.5 Delayed FlexRay transceiver deactivation at Shutdown

The transition (T11) from FRSM_HALT_REQ to FRSM_READY may be delayed by a configured timer value see 3.3.8.

7 Glossary and Abbreviations

7.1 Glossary

Term	Description
DaVinci Configurator	Generation tool for MICROSAR components
FIBEX	Field Bus EXchange format a bus system specific container for PDUs
Cold-Start node	ECU has an active part at the timing synchronization at Startup
Wake-up node	ECU which has the permission to wake up a FlexRay channel by sending a WUP

Table 7-1 Glossary

7.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
BswM	Basic SW Manager
CC	Communication Controller
Cdd	Complex Device Driver
Cfg	configuration
ComM	Communication Manager
Dem	Diagnostic Event Manager
Det	Development Error Tracer
ECU	Electronic Control Unit
FullCom	`COMM_FULL_COMMUNICATION`
Fr	FlexRay
FrIf	FlexRay Interface
FrSM	FlexRay State Manager
FrNm	FlexRay Network Management
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
NoCom	`COMM_NO_COMMUNICATION`
OS	Operating System
POC	Protocol Operation Control
SchM	BSW Schedule Manager
SW	Soft Ware
SWC	Software Component
WU	Wake-Up

WUP	Wake-Up Pattern
WUS	Wake-Up Sequence

Table 7-2 Abbreviations

8 Contact

Visit our website for more information on

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

www.vector.com