BMW GROUP

Rolls-Royce Motor Cars Limited

# Coding Classic Integration Manual

| | |
|---|---|
| Project | BMW AUTOSAR 4 Core Rel. 3 |
| Author | BMW AG |
| Release Date | 2017-12-14 |
| Version | 5.2.1 |
| Status | Release |
| Hotline | +49 89 382 - 32233 |
| Contact | bac@bmw.de<br>https://asc.bmw.com/jira/browse/BSUP (extern)<br>https://asc.bmwgroup.net/jira/browse/BSUP (intern) |

## Revision History

| Version | Date | Description |
|---|---|---|
| 5.2.1 | 2017-12-14 | Version Update |
| 5.2.0 | 2017-11-09 | Version Update |
| 5.1.0 | 2017-10-12 | Version Update |
| 5.0.1 | 2017-09-14 | Version Update |
| 5.0.0 | 2017-06-08 | Initial version for SP2021 |

**Company**
Bayerische Motoren Werke Aktiengesellschaft
**Postal address**
BMW AG
80788 München
**Office address**
Forschungs- und Innovationszentrum (FIZ)
Hufelandstr. 1
80937 München
**Telephone**
Switchboard
+49 89 382-0
**Internet**
www.bmwgroup.com

BMW GROUP

Rolls-Royce Motor Cars Limited

# Table of Contents

# 1 Introduction

The Coding system consists of a Coding module that is part of the ECU application and the Coder.

Almost in the same manner as the Bootloader programs the ROM of an ECU, the Coding module codes the NV memory of an ECU with vehicle specific data derived from the vehicle description document e.g. type key, extra equipment, country specifics, year of model/construction stage, additional comments or service upgrade options.

## General

For a general introduction to the BAC4/aBAC Modules please refer to [1].

This document only describes topics related to the Coding BAC4/aBAC Module.

This Integration Manual describes the basis functionality, API and the configuration of the BMW system function Coding.

## Functional overview

The Coding module implements the coding process that is part of the vehicle programming process. It provides functions to the application to read coded data stored in the NV memory. If the NV memory contains no valid data default values are provided. The coded data in NV memory can only be changed via a diagnostic coding session.

The Coding module provides the following functionalities:

- Manages coding data stored in NV memory
- Provides read/write access via a special UDS diagnostics Session to the Coder
- Provides read only access via API to the ECU application
- Informs the ECU application about its current status
- Performs signature checks over the Coding data
- Restores safe default values in case of errors

# 2 Related documentation

## References

[1] BAC4 General Concept for the Module Integration
BAC4_General_Concepts_for_the_Module_Integration.pdf

# 3 Limitations

– The Coding module supports only one asymmetric key to calculate the signature for all Coding areas.

# 4 Software Architecture

## Dependencies on AUTOSAR modules

The current version of the Module Coding depends on the following BSW modules:

### NvM

The Coding module uses the NVM to read Coding data from NV memory and to write it back.

### Dem

The Coding module uses the DEM API to report critical events and write error memory entries.

### Dcm

The Coding module uses the DCM to communicate with the Coder by sending and receiving diagnostic UDS messages via CAN, FlexRay, etc.

### Com

The Coding module use signals from COM to get the Vehicle Speed.

### Det

The Coding module optionally reports development errors to the Det.

### RTE

As a software component, the Coding module uses Rte client/server and sender/receiver communication to communicate with other SWCs and BSW modules.

### BswM

The Coding receives and requests mode switches from the BswM to switch the current Coding operational mode. It further receives a mode switch from the BswM when the active diagnostic session has changed and when the bus communication status has changed.

## Dependencies to other modules

### Dlog

The Dlog module is used to obtain the information during startup if the control unit has been coded after flash programming.

### Vin

The Vin module is used to obtain the current bus VIN.

### Other SWC

The Coding optionally calls another Application Software Component to perform a plausibility check of the net coding data received during the coding data transaction if implausible net coding data can lead to safety-critical states of the control unit.

# 5 Integration

## Configuration of other Modules

The following modules shall be configured, before this module can be generated, compiled and linked.

### NvM

The NVRAM Manager shall be configured to hold one NVRAM block. The following parameters have to be configured:

- a container "'NvMBlockDescriptor"' with
  - NvmNvBlockLength: "'see CODING_DATA_SIZE in generated Coding_Cfg.h"'
  - NvmBlockCRCType: e.g. NVM_CRC16
  - NvMBlockUseCrc: true
  - NvMCalcRamBlockCrc: false
  - NvmBlockManagementType: NVM_BLOCK_NATIVE
  - NvmRamBlockDataAddress: &Coding_CodingData
  - NvmRomBlockDataAddress: &Coding_CodingDataDefault
  - NvmSelectBlockForReadAll: true
  - NvmSelectBlockForWriteAll: false

This NVRAM block needs

- Port and Port Interface for service ports
- Port and Port Interface for single block job end notifications

This NVRAM block will be used to store Coding data and will be written on Coding process.

### Dem

The Coding module reports different errors to the DEM. The DEM shall be configured to provide the following Errors (aging/healing is disabled for all DTCs):

- CODING_EVENT_NOT_CODED
  - Description: No valid Coding data is available and default values are used.
  - Annotation: A Clear Event Allowed port shall be configured via Rte
  - DTC Class
    - DemUdsDtc: 0x20000 + 0x100 * ECU_Address + 0x08
    - DemDTCSeverity: DEM_DTC_SEV_NO_SEVERITY
  - Event
    - DemEventKind: DEM_EVENT_KIND_SWC
    - DemEventDestination: DEM_DTC_ORIGIN_PRIMARY_MEMORY
- CODING_EVENT_WRONG_VEHICLE
  - Description: CPS and VIN do not match.
  - DTC Class
    - DemUdsDtc: 0x20000 + 0x100 * ECU_Address + 0x09

    - DemDTCSeverity: DEM_DTC_SEV_NO_SEVERITY
  - Event
    - DemEventKind: DEM_EVENT_KIND_SWC
    - DemEventDestination: DEM_DTC_ORIGIN_PRIMARY_MEMORY
- CODING_EVENT_UNQUALIFIED_DATA
  - Description: CPS-VIN comparison has not been done.
  - Annotation: Error entry depends on configuration parameter "'CodingProdErrorCEUDDetection"'
  - DTC Class
    - DemUdsDtc: 0x20000 + 0x100 * ECU_Address + 0x0A
    - DemDTCSeverity: DEM_DTC_SEV_NO_SEVERITY
  - Event
    - DemEventKind: DEM_EVENT_KIND_SWC
    - DemEventDestination: DEM_DTC_ORIGIN_PRIMARY_MEMORY

### Dcm

### DiagnosticSession

The writing of net coding data into a control unit must only be possible, if the control unit is in the CodingSession.

The following parameters have to be configured:

- a container "'DcmDspSessionRow"' with
  - DcmDspSessionForBoot: DCM_NO_BOOT
  - DcmDspSessionLevel: 65 (0x41)
  - DcmDspSessionP2ServerMax: **[IM_CodingClassic_001]** ⌈⌋(DK_T3_266)
  - DcmDspSessionP2StarServerMax: 5.0

### DiagnosticSessionControl

**[IM_CodingClassic_002]** ⌈The Diagnostic Session Control service is used for changing codable control units into the Extended Diagnostic Session, the Coding Session and the Default Session. ⌋(COD_268)

**[IM_CodingClassic_003]** ⌈The following parameters have to be configured: ⌋(COD_269, COD_272, COD_276, COD_280)

- a container "'DcmDsdService"' with
  - DcmDsdSidTabServiceId: 0x10
    - a container "'DcmDsdSubService"' with
      - DcmDsdSubServiceId: 0x01
      - DcmDsdSubServiceSessionLevelRef: DEFAULT_SESSION
      - DcmDsdSubServiceSessionLevelRef: EXTENDED_SESSION
      - DcmDsdSubServiceSessionLevelRef: CODING_SESSION
      - DcmDsdSubServiceId: 0x03
      - DcmDsdSubServiceSessionLevelRef: DEFAULT_SESSION
      - DcmDsdSubServiceSessionLevelRef: EXTENDED_SESSION

- DcmDsdSubServiceId: 0x41
- DcmDsdSubServiceSessionLevelRef: EXTENDED_SESSION
- DcmDsdSubServiceSessionLevelRef: CODING_SESSION

**RoutineControl**

**[IM_CodingClassic_004]** ⌈The Routine Control service is used for reading, writing and verifying net coding data. ⌋(COD_289)

**Reading of net coding data from control unit (0x31 0x01 0x37 0xFC)**

**[IM_CodingClassic_005]** ⌈This request has a variable response length. The following parameters have to be configured: ⌋(COD_292, COD_294)

- a container "'DcmDspRoutine"' with
  - DcmDspRoutineIdentifier: 0x37FC
  - DcmDspRoutineUsePort: TRUE

- a container "'DcmDspRoutineInfo"' and "'DcmDspStartRoutineIn"' with
  - a "'StartRoutineInSignal"' with
    - DcmDspRoutineSignalLength: 48 bits
    - DcmDspRoutineSignalPos: 0
    - DcmDspRoutineSignalType: UINT8
  - a "'StartRoutineOutSignal"' with
    - DcmDspRoutineSignalLength: max. diagnostic buffer size (e.g. 8192 bits)
    - DcmDspRoutineSignalPos: 0
    - DcmDspRoutineSignalType: VARIABLE_LENGTH

**Writing of net coding data from control unit (0x31 0x01 0x37 0xFD)**

**[IM_CodingClassic_006]** ⌈This request has a variable request length. The following parameters have to be configured: ⌋(COD_297, COD_299)

- a container "'DcmDspRoutine"' with
  - DcmDspRoutineIdentifier: 0x37FD
  - DcmDspRoutineUsePort: TRUE

- a container "'DcmDspRoutineInfo"' and "'DcmDspStartRoutineIn"' with
  - a "'StartRoutineInSignal"' with
    - DcmDspRoutineSignalLength: max. diagnostic buffer size (e.g. 8192 bits)
    - DcmDspRoutineSignalPos: 0
    - DcmDspRoutineSignalType: VARIABLE_LENGTH

### Verification of net coding data from control unit (0x31 0x01 0x37 0xFE)

**[IM_CodingClassic_007]** ⌈The following parameters have to be configured: ⌋(COD_302, COD_304)

- a container "'DcmDspRoutine"' with
  - DcmDspRoutineIdentifier: 0x37FE
  - DcmDspRoutineUsePort: TRUE

- a container "'DcmDspRoutineInfo"' and "'DcmDspStartRoutineIn"' with
  - a "'StartRoutineInSignal"' with
    - DcmDspRoutineSignalLength: 144 bits
    - DcmDspRoutineSignalPos: 0
    - DcmDspRoutineSignalType: UINT8
  - a "'StartRoutineOutSignal"' with
    - DcmDspRoutineSignalLength: 8 bits
    - DcmDspRoutineSignalPos: 0
    - DcmDspRoutineSignalType: UINT8

### ReadDataByIdentifier

**[IM_CodingClassic_008]** ⌈The Read Data By Identifier service is used in conjunction with coding for the following request to the control unit:

- Determining the value of the coding proof stamp (CPS)
- Determining the current session status (reading ActiveSessionState)
- Determining the current session
- Determining the protocol data (transport buffer sizes, coding area count and lengths)

⌋(COD_289)

### Read coding proof stamp (CPS) (0x22 0x37 0xFE)

Reading out the coding proof stamp (CPS) from the control unit.

**[IM_CodingClassic_009]** ⌈The following parameters have to be configured: ⌋(COD_314, COD_316)

- a container "'DcmDspDataInfo"' with
  - DcmDspDataFixedLength: TRUE

- a container "'DcmDspData"' with
  - DcmDspDataConditionCheckReadFncUsed: TRUE
  - DcmDspDataInfoRef: reference to the "'DcmDspDataInfo"' container configured before.
  - DcmDspDataSize: 144 bits
  - DcmDspDataType: UINT8_N
  - DcmDspDataUsePort: USE_DATA_SYNCH_CLIENT_SERVER

- a container "'DcmDspDidInfo"' with
  - only read-access

  - DcmDspDidReadSessionRef: CODING_SESSION, EXTENDED_SESSION, DEFAULT_SESSION

- a container "'DcmDspDid"' with
  - DcmDspDidIdentifier: 0x37FE
  - one DID Signal "'DcmDspDidSignal"' with Data Position = 0 and a Data Reference to the "'DcmDspData"' container configured before.

**Read status of the active Session (0x22 0xF1 0x00)**

Reading out the status of the ActiveSessionState from the control unit.

**[IM_CodingClassic_010]** ⌈The following parameters have to be configured: ⌋(COD_319, COD_321)

- a container "'DcmDspDataInfo"' with
  - DcmDspDataFixedLength: TRUE

- a container "'DcmDspData"' with
  - DcmDspDataConditionCheckReadFncUsed: TRUE
  - DcmDspDataInfoRef: reference to the "'DcmDspDataInfo"' container configured before.
  - DcmDspDataSize: 32 bits
  - DcmDspDataType: UINT8_N
  - DcmDspDataUsePort: USE_DATA_SYNCH_CLIENT_SERVER

- a container "'DcmDspDidInfo"' with
  - only read-access
  - DcmDspDidReadSessionRef: CODING_SESSION, EXTENDED_SESSION, DEFAULT_SESSION

- a container "'DcmDspDid"' with
  - DcmDspDidIdentifier: 0xF100
  - one DID Signal "'DcmDspDidSignal"' with Data Position = 0 and a Data Reference to the "'DcmDspData"' container configured before.

**Read active session (0x22 0xF1 0x86)**

Reading out of the ActiveSession from the control unit.

**[IM_CodingClassic_011]** ⌈The following parameters have to be configured: ⌋(COD_324, COD_326)

- a container "'DcmDspDataInfo"' with
  - DcmDspDataFixedLength: TRUE

- a container "'DcmDspData"' with
  - DcmDspDataConditionCheckReadFncUsed: TRUE
  - DcmDspDataInfoRef: reference to the "'DcmDspDataInfo"' container configured before.
  - DcmDspDataSize: 8 bits
  - DcmDspDataType: UINT8_N
  - DcmDspDataUsePort: USE_DATA_SYNCH_FNC

- a container "'DcmDspDidInfo"' with

  - only read-access
  - DcmDspDidReadSessionRef: CODING_SESSION, EXTENDED_SESSION, DEFAULT_SESSION

- a container "'DcmDspDid"' with
  - DcmDspDidIdentifier: 0xF186
  - one DID Signal "'DcmDspDidSignal"' with Data Position = 0 and a Data Reference to the "'DcmDspData"' container configured before.

**Read protocol data (0x22 0x37 0xFF)**

Reading of the protocol data from the control unit.

**[IM_CodingClassic_012]** ⌈The following parameters have to be configured: ⌋(COD_329, COD_331)

- a container "'DcmDspDataInfo"' with
  - DcmDspDataFixedLength: TRUE

- a container "'DcmDspData"' with
  - DcmDspDataConditionCheckReadFncUsed: TRUE
  - DcmDspDataInfoRef: reference to the "'DcmDspDataInfo"' container configured before.
  - DcmDspDataSize: 72 bits
  - DcmDspDataType: UINT8_N
  - DcmDspDataUsePort: USE_DATA_SYNCH_CLIENT_SERVER

- a container "'DcmDspDidInfo"' with
  - only read-access
  - DcmDspDidReadSessionRef: CODING_SESSION, EXTENDED_SESSION, DEFAULT_SESSION

- a container "'DcmDspDid"' with
  - DcmDspDidIdentifier: 0x37FF
  - one DID Signal "'DcmDspDidSignal"' with Data Position = 0 and a Data Reference to the "'DcmDspData"' container configured before.

**DiagnosticSessionControl**

**[IM_CodingClassic_013]** ⌈The ECUReset service is used in order to prompt a control unit to exit the CodingSession following successfully completed coding. ⌋(COD_338)

**[IM_CodingClassic_014]** ⌈The following parameters have to be configured: ⌋(COD_341, COD_343)

- a container "'DcmDsdService"' with
  - DcmDsdSidTabServiceId: 0x11
    - a container "'DcmDsdSubService"' with
      - DcmDsdSubServiceId: 0x01
      - DcmDsdSubServiceSessionLevelRef: DEFAULT_SESSION
      - DcmDsdSubServiceSessionLevelRef: EXTENDED_SESSION
      - DcmDsdSubServiceSessionLevelRef: CODING_SESSION

### Service Request Manufacturer Notification

A Coding Service Request Manufacturer Notification shall be configured in the container "DcmDslServiceRequestManufacturerNotification". The Mapping to the Coding module is done by connecting the corresponding ports.

### Com

The communication stack has to be configured for the reception and sending of bus messages. The following PDUs/Signals are expected to be present in the Com stack after import of the BMW BNE export (CodingSample):

- Speed Signal (Rx) - CAN
  - Shortname: V_VEH
  - Length: 5 Bytes
  - Signal: V_VEH_COG
  - Signal Size: 16 Bits
  - Startbit Position: 16
- Speed Signal (Rx) - Ethernet
  - Service Interface ID: 0x7533
  - Service: SpeedAcceleration
  - Ereignis/Field: VehicleSpeedAndDrivingCondition
  - Parameter/Strukturelement: speedVehicleCentreOfGravity

Important Note: For integration of the CodingSample module in an ECU, which is connected to Ethernet as "'System Bus"' there is an adaption to make (S/R-interface and code). A new implementation data type has to be created e.g. "'CodingSample_VehicleSpeedAndDrivingRecordType"' and the existing S/R-interface "'Coding_VehicleSpeedInterface"' must refer this new data implementation type. Furthermore the provided function "'CodingSample_VehicleSpeedReceiveHandler()"' must be modified to access the desired parameter of the new structure.

### Det

The DET shall be configured to provide development Errors. These errors are application programmer specific.

### BswM

The BswM has to provide:

- BswMModeRequestPort "'LifeCycle"' to receive mode switch notifications of the ModeDeclarationGroup "'Coding_LifeCycle"' from Coding
- BswMRteModeRequestPort "'LifeCycleRequest"' to request modes of the ModeDeclarationGroup "'Coding_LifeCycle"'
- BswMRules to request the Coding life cycle mode
- BswMRules to switch the Coding BusCom-Mode

– BswMRules to switch the Coding Session-Mode

## Configuration of generic part

### CodingCrypto

This container contains the configuration (parameters) of the crypto part of the Coding module.

### CryptoLib

Choose this container to use the BMW crypto library for crypto operations.

### Csm

Choose this container to use the AUTOSAR Csm for crypto operations.

### CodingHashSize

This parameter defines the size of the hash depending on the Hash Method.

The Coding module supports the following hash algorithm

– SHA-256 - length 32 bytes
– SHA-384 - length 43 bytes
– SHA-512 - length 64 bytes

### CodingGeneral

This container contains the configuration parameters of the generic part of the Coding module.

### CodingDevErrorDetect

This parameter activates/deactivates the Development Error Detection and Notification.

If set to true: Development Error Detection and Notification is activated.
If set to false: Development Error Detection and Notification is deactivated.

### CodingSignatureSize

**[IM_Coding_001]** ⌈This parameter defines the size of the net Coding Data Signature depending on the Signature Method. ⌋(COD_147, COD_148)

The Coding module supports the following asymmetric keys

- RSA key length 2048 bit
- RSA key length 3072 bit
- RSA key length 4096 bit
- ECC key length 256 bit
- ECC key length 384 bit
- ECC key length 521 bit

### CodingCryptoEnable

This parameter activates/deactivates the Service to check net Coding Data Signature.

If set to true: Service to check net Coding Data Signature is activated.
If set to false: Service to check net Coding Data Signature is deactivated.

### CodingProdErrorCEUDDetection

**[IM_Coding_002]** ⌈This parameter activates/deactivates the Production Error "'Coding Event Unqualified Data"' (CEUD) Detection and Reporting. ⌋(COD_114, COD_138)

If set to true: Production Error CEUD Detection and Report is activated.
If set to false: Production Error CEUD Detection and Report is deactivated.

### CodingConditionCheck

**[IM_Coding_003]** ⌈This parameter activates/deactivates the Coding Ability due to Operating Conditions Detection e.g. car is being driven or engine is running. ⌋(COD_060)

If set to true: Operating Conditions Detection is activated.
If set to false: Operating Conditions Detection is deactivated.

### CodingReceiveBufferSize

**[IM_Coding_004]** ⌈This parameter defines the size of the Coding Receive Buffer (Rx) of Control Unit in Bytes. ⌋(COD_171)

### CodingSendBufferSize

**[IM_Coding_005]** ⌈This parameter defines the size of the Coding Send Buffer (Tx) of Control Unit in Bytes. ⌋(COD_171)

### CodingArea

**[IM_Coding_006]** ⌈This container contains the default value for unused bits and all user configurable functions of this Coding area. ⌋(COD_097)

### CodingAreaDefVal

This parameter defines the Default Value for Unused Bits in the Coding Area.

The Coding module supports the following default values

- AllBitsZero (0x00)
- AllBitsOne (0xFF)

### CodingFunction

This container contains one or more Coding Functions.

### CodingFunctionStartByte

This parameter contains the Startbyte of the Coding Function Value within the User Configurable Data.

### CodingFunctionEndByte

This parameter contains the Endbyte of the Coding Function Value within the User Configurable Data.

### CodingFunctionMask

This parameter contains the Mask of the Coding Function Value within the User Configurable Data.

### CodingFunctionApplType

This parameter defines the Function Type of the Coding Function Value towards the Application.

The Coding module supports the following types

- boolean
- bytearray
- uint8
- uint16
- uint32
- uint64
- sint8
- sint16
- sint32
- sint64
- float32
- float64

**CodingFunctionDataCheck**

**[IM_Coding_007]** ⌈This parameter activates/deactivates the Coding Function Value Check. Control Units must perform a Plausibility Check of the Net Coding Data received during the Coding Data Transaction if implausible Net Coding Data can lead to safety-critical states of the Control Unit. ⌋(COD_227)

If set to true: Coding Function Value Check is activated.
If set to false: Coding Function Value Check is deactivated.

**CodingFunctionTransformationRule**

This parameter defines the Transformation Rule for the Coding Function Value from Non Volatile Memory to the User Interface (Application). This Rule shall be configured as a Simple C Operation e.g. "'+10"' or "'*2"'.

**CodingFunctionLowerLimit**

This parameter defines the Lower Limit for the Coding Function Value passed to the User Interface (Application).

**CodingFunctionUpperLimit**

This parameter defines the Upper Limit for the Coding Function Value passed to the User Interface (Application).

**CodingFunctionNvramType**

This container defines the Function Type of the Coding Function Value in Non Volatile Memory.

The Coding module supports the following types

- boolean
- bytearray
- uint8
- uint16
- uint32
- uint64
- sint8
- sint16
- sint32
- sint64
- float32
- float64

Choose the desired container to defines the Function Type of the Coding Function Value in Non Volatile Memory

CodingFunctionDefaultValue

This parameter contains the Default Value of the Coding Function within the User Configurable Data.

## Configuration of adapter part

### CodingClassicGeneral

This container contains the configuration parameters of the classic adapter of the Coding module.

### CodingCycleTime

This parameter defines the Call Cycle of the Coding Timer Function in Seconds.

### Other Application SWC

- Application Connectors
  - CheckCodingFunction_<x>: Shall be connected with its corresponding CodingSample ports.
  - ConditionMode: Shall be connected with its corresponding CodingSample port.
  - Data: Shall be connected with its corresponding CodingSample and CodingTestClient port.
  - DataMode: Shall be connected with its corresponding CodingSample and CodingTestClient port.
  - LifeCycle: Shall be connected with its corresponding BswM and CodingTestClient port.
  - ProgId: Shall be connected with its corresponding Dlog port.
  - SVK: Shall be connected with its corresponding Dlog port.
  - Vin: Shall be connected with its corresponding Vin port.
- Service Connectors
  - BusCom: Shall be connected with its corresponding BswM port.
  - CBClrEvt_EventNotCoded: Shall be connected with its corresponding Dem port.

- CheckNCD: Shall be connected with its corresponding Dcm port.
- DETService: Shall be connected with its corresponding Det port.
- EventNotCoded: Shall be connected with its corresponding Dem port.
- EventUnqualifiedData: Shall be connected with its corresponding Dem port.
- EventWrongVehicle: Shall be connected with its corresponding Dem port.
- LifeCycleRequest: Shall be connected with its corresponding BswM and CodingTestClient port.
- Notification: Shall be connected with its corresponding Dcm port.
- NvMNotifyJobFinished: Shall be connected with its corresponding NvM port.
- NvMService: Shall be connected with its corresponding NvM port.
- ReadCPS: Shall be connected with its corresponding Dcm port.
- ReadNCD: Shall be connected with its corresponding Dcm port.
- ReadProtocolData: Shall be connected with its corresponding Dcm port.
- SessionChangeIndication: Shall be connected with its corresponding BswM port.
- WriteNCD: Shall be connected with its corresponding Dcm port.

### Event Mapping

- Coding_DemClearEventAllowedNotCoded: Operation Invoked Event
- Coding_TimerMain: Timing Event
- DCMConfirmation: Operation Invoked Event
- DCMIndication: Operation Invoked Event
- GetCafIds: Operation Invoked Event
- LifeCycleHandler: Data Receive Event
- R_CheckNCD: Operation Invoked Event
- R_ConditionCheckReadCPS: Operation Invoked Event
- R_ConditionCheckReadProtocolData: Operation Invoked Event
- R_GetCodingFunction_<x>: Operation Invoked Event
- R_ReadCPS: Operation Invoked Event
- R_ReadNCD: Operation Invoked Event
- R_ReadProtocolData: Operation Invoked Event
- R_WriteNCD: Operation Invoked Event
- SessionChange: Mode Switch Event
- VinReceiveHandler: Data Receive Event
- WriteDataBlockNotification: Operation Invoked Event

### Data Mapping

No Data Mapping necessary for the Coding module.

### Exclusive Areas

The exlusive areas CodingData and CodingState shall be configured.

## Software Integration

### Startup/Initialization

Before the Coding operation mode CODING_INITIALIZED is requested, the following preconditions shall be satisfied:

- The Coding module is in STOPPED-Mode
- The NVRAM Manager is initialized
- The NVRAM Manager has finished the job Nvm_ReadAll(). This implies that the NV block containing the Coding data has also been loaded. If the block is corrupted, the NVRAM Manager automatically loads the default ROM block containing the default values.
- The Dlog is in RUNNING-Mode

The integrator has to assure, that the Coding operation mode CODING_INITIALIZED is requested, after NvM_ReadAll() is finished and Dlog module switched to RUNNING-Mode.

### Normal Operation

Before the Coding operation mode CODING_RUNNING is requested, the following preconditions shall be satisfied:

- The Coding module is in INITIALIZED-Mode

The Coding_TimerMain() will be scheduled cyclically via RTE (a good value for this is a 100ms cycle ). After the Coding BusCom-Mode is set to CODING_BUSCOMON the Coding_TimerMain() may request the Vin module to get the onboard electrical system VIN to validate net Coding data if no VIN has been received until now. This qualification process is repeated 3 times at 1000 ms intervals, if a VIN has still not been received from the onboard electrical system. After this, qualification of the net coding data is not performed in this wake cycle, even if an onboard electrical system VIN is received at a later point.

### Shutdown/Deactivation

Before the Coding operation mode CODING_STOPPED is requested, the following preconditions shall be satisfied:

- The Coding module is either in INITIALIZED- or RUNNING-Mode
- Shutdown/Deactivation is requested

The integrator has to assure, that the Coding operation mode CODING_STOPPED is requested, after a shutdown/deactivation request was received.

### SWCD

The Coding module provides the following Mode:
ModeDeclarationGroup Coding_ConditionMode

{
{
CODING_ALLOWED
CODING_NOT_ALLLOWED
}
initialMode = CODING_ALLOWED
}
This mode is used to enable or disable the start of a Coding process. An implementation how to change this Mode is done in CodingSample_VehicleSpeedReceiveHandler(). The user can add additional rules.

The Coding module provides the following Mode:
ModeDeclarationGroup Coding_BusComMode
{
{
CODING_BUSCOMOFF,
CODING_BUSCOMON
}
initialMode = CODING_BUSCOMOFF
}
The integrator has to assure, that this Mode is set to CODING_BUSCOMON, after the bus communication has started. The Coding module requests the VIN from the Vin module after this Mode is set to CODING_BUSCOMON.

No special callback functions have to be introduced by the application programmer for using the Coding module. The following application callback functions are already implemented in CodingSample.c and can be adapted by the application programmer:

- CodingSample_VehicleSpeedReceiveHandler()
- CodingSample_CheckCodingFunction_<x>()