Document Title	Specification of Communication
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	015
Document Classification	Standard

Document Version	2.3.0
Document Status	Final
Part of Release	2.1
Revision	20

Document Change History

Date	Version	Changed by	Change Description
17.05.2010	2.3.0	AUTOSAR Administration	• Updated COM222 of release 2.1 as COM222 in release 3.X/4.X with respect to the update-bit clearing • Tables were wrongly stating that Com_ReceiveShadowSignal should return COM_SERVICE_NOT_AVAILABLE. • Figure 19 misses a link between the COM_SIGNAL_GROUP container and the COM_NOTIFICATION_ERROR container • Corrections to non-normative overview figure • Added the missing word "notifications" in COM053 • Rephrase COM287 to replace the misleading term "normal signal indication" • Legal disclaimer revised
06.05.2008	2.2.0	AUTOSAR Administration	• Remove limitation of 8 Byte for I-PDUs sent on FlexRay • updated chapter 4.1 • updated COM176 • Legal disclaimer revised

30.01.2007	2.1.0	AUTOSAR Administration	Clarifications of requirements throughout the whole document. Chapter 11 contains a detailed list of the made modifications. No new major features were added since release 2.0. • "Advice for users" revised • "Revision Information" added • Legal disclaimer revised
21.04.2006	2.0.0	AUTOSAR Administration	Document structure adapted to common Release 2.0 SWS Template. • Integration of Signal Gateway • Major updates in configuration, error handling, filtering, Transmission Mode Switches, callouts, Update-bits, dead-line monitoring and initialization
31.05.2005	1.0.0	AUTOSAR Administration	Initial release

Disclaimer

Advice for users

AUTOSAR specifications may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Introduction and functional overview

This specification is the AUTOSAR COM Software Specification. It is based on the AUTOSAR COM SRS [7]. It specifies how the requirements of the AUTOSAR COM SRS shall be realized. That means that the functionality and the API of the AUTOSAR COM module are described in this document.

Within the AUTOSAR Layered Architecture the COM module is placed between RTE and the PDU Router, see [1].

AUTOSAR COM is derived from [17]. For details see Chapter 7.2.1.
AUTOSAR COM provides signal gateway functionality. For details see Chapter 7.2.4.

Main Features:

- Provision of signal oriented data interface for the RTE
- Packing of AUTOSAR signals to I-PDUs to be transmitted
- Unpacking of received I-PDUs and provision of received signals to RTE
- Routing of signals from received I-PDUs into I-PDUs to become transmitted
- Routing of signal groups from received I-PDUs into I-PDUs to become transmitted
- Communication transmission control (start/stop of I-PDU groups)
- Replications of send requests
- Guarantee of minimum distances between transmit I-PDUs
- Monitoring of receive signals (signals timeout)
- Filter mechanisms for incoming signals
- Different notification mechanisms
- Provision of Init-Values and Update-Indications
- Byte order conversion
- Sign extension
- Support of two different Transmission Modes per I-PDU
- Signal based Gateway

2 Acronyms, abbreviations and definitions

2.1 Acronyms and abbreviations

Acronym:	*Description:*
AUTOSAR COM	AUTOSAR COM is derived from OSEK COM [17]. For details see Chapter 7.2.1.
DM	Deadline Monitoring. For details see Chapter 7.4.5.4.
IL	Interaction Layer. A detailed description can be found in [17].
I-PDU	Interaction Layer Protocol Data Unit An I-PDU carries signals and is defined in [17].
LOM	Listen only mode
L-PDU	Data Link Layer PDU. In AUTOSAR the Data Link Layer is equivalent to the Communication Hardware Abstraction and Microcontroller Abstraction Layer.
OSEK COM	Open systems and the corresponding interfaces for automotive electronics – communication [17]
PCI	Protocol Control Information For a description see [1] page 04-51 ff.
PDU Router	Module that transfers I-PDUs from one module to another module. The PDU Router can be utilized for gateway operations and for internal routing purposes.
SDT	Specification of System Template [12]
SDU	Service Data Unit For a description see [1] Chapter 4.
TM	Transmission Mode
TMC	Transmission Mode Condition (see Chapter 7.4.3.3)
TMS	Transmission Mode Selector (see Chapter 7.4.3.3)

2.1.1 Definitions

Acronym:	*Description:*
Confirmation	Lower layer reports that a request by COM has been completed successfully. It's a reaction to a request of COM. E.g. when a PDU has been successfully transmitted.
Data Invalid Value	Value sent by the AUTOSAR COM to indicate that the sender side AUTOSAR Software Component is not able to provide a valid value.
Indication	Asynchronous information from lower layer to COM, e.g. when something has been received.
Init Value	I-PDUs and signals are set to the Init Value by AUTOSAR COM after start-up. This value is used until it is overwritten.
I-PDU group	An I-PDU group contains zero or more I-PDUs or I-PDU groups. For a detailed description see [7].
Inter-ECU communication	Communication between two or more ECU for example via a CAN network.
Intra-ECU communication	Communication between Software components that reside on the same ECU.
Message	OSEK-COM uses always the synonym *message*. In AUTOSAR, *message* is replaced by *signal* but with the same meaning.
Network signal	A network signal is the representation of a COM signal on the bus. For the configuration of network signals see 10.2.2.8.
Notification	Information by COM to upper layer, e.g. when new data is available, an error occurred.
Signal	A description can be found in [7].

Acronym:	*Description:*
Signal groups	In AUTOSAR, so called *complex data types* are used. Inside a complex data type, there are one or more data elements (primitive data types), like in a C struct. To ensure data consistency a complex data type must be treated as an atomic unit. The RTE decomposes the complex data type in single signals and sends them to the AUTOSAR COM module. As these signals altogether still have to be treated as atomic, they are called *signal group*. See also [7].
Update-bits	A mechanism supported by AUTOSAR COM with that the receiver of a signal/ signal group can identify whether the sender has updated the data in this signal/ signal group before sending. See Chapter 7.7.

3 Related documentation

3.1 Deliverables of AUTOSAR

[1] Layered Software Architecture
AUTOSAR_LayeredSoftwareArchitecture.pdf

[2] Specification of Communication Stack Types
AUTOSAR_SWS_ComStackTypes.pdf

[3] General Requirements on Basic Software Modules
AUTOSAR_SRS_General.pdf

[4] Basic Software UML Model
AUTOSAR_UML_Model.eap

[5] Specification of Standard Types
AUTOSAR_SWS_StandardTypes.pdf

[6] Specification of the Virtual Functional Bus
AUTOSAR_VirtualFunctionBus.pdf

[7] Requirements on Communication
AUTOSAR_SRS_COM.pdf

[8] Software Component Template
AUTOSAR_SoftwareComponentTemplate.pdf

[9] Requirements on Gateway
AUTOSAR_SRS_Gateway.pdf

[10] Specification of PDU Router
AUTOSAR_SWS_PDU_Router.pdf

[11] Specification of Operating System
AUTOSAR_SWS_OS.pdf

[12] Specification of System Template
AUTOSAR_SystemTemplate.pdf

[13] Specification of RTE Software
AUTOSAR_SWS_RTE.pdf

[14] Specification of ECU Configuration
AUTOSAR_ECU_Configuration.pdf

[15] Specification of Generic Network Management
AUTOSAR_SWS_Generic_NM.pdf

[16] Specification of Communication Manager
AUTOSAR_SWS_ComManager.pdf

3.2 Related standards and norms

[17] OSEK/ VDX Communication Version 3.0.3
http://www.osek-vdx.org
OSEKCOM303.pdf

[18] OSEK implementation language Version 2.5
http://www.osek-vdx.org
oil25.pdf

4 Constraints and assumptions

This document is applicable for AUTOSAR release 2.1. Features for later AUTOSAR releases are not yet included.

4.1 Limitations

Within the AUTOSAR COM Stack the I-PDUs of COM are passed via the PDU Router directly to the communication interfaces. Therefore the maximum length of an I-PDU depends on the maximum length of the L-PDU of the underlying communication interface. For CAN and LIN the maximum L-PDU length is 8 bytes. For FlexRay the maximum L-PDU length is 254 bytes.

AUTOSAR COM is based on [17]. Nevertheless not all features of [17] are included and some features are different. See COM013 for a list of not included features.

For AUTOSAR release 2.1 there is no special treatment in COM to handle *floating point* data types. However they may be handled with existing COM functionality.

The number of IPDU-groups is limited to 32, see COM187.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to other modules

This chapter lists all the features from other modules which are used by the AUTSAR COM module and functionalities which are provided by AUTOSAR COM to other modules. For the placement of AUTOSAR COM in the communication stack see **Figure 1**.

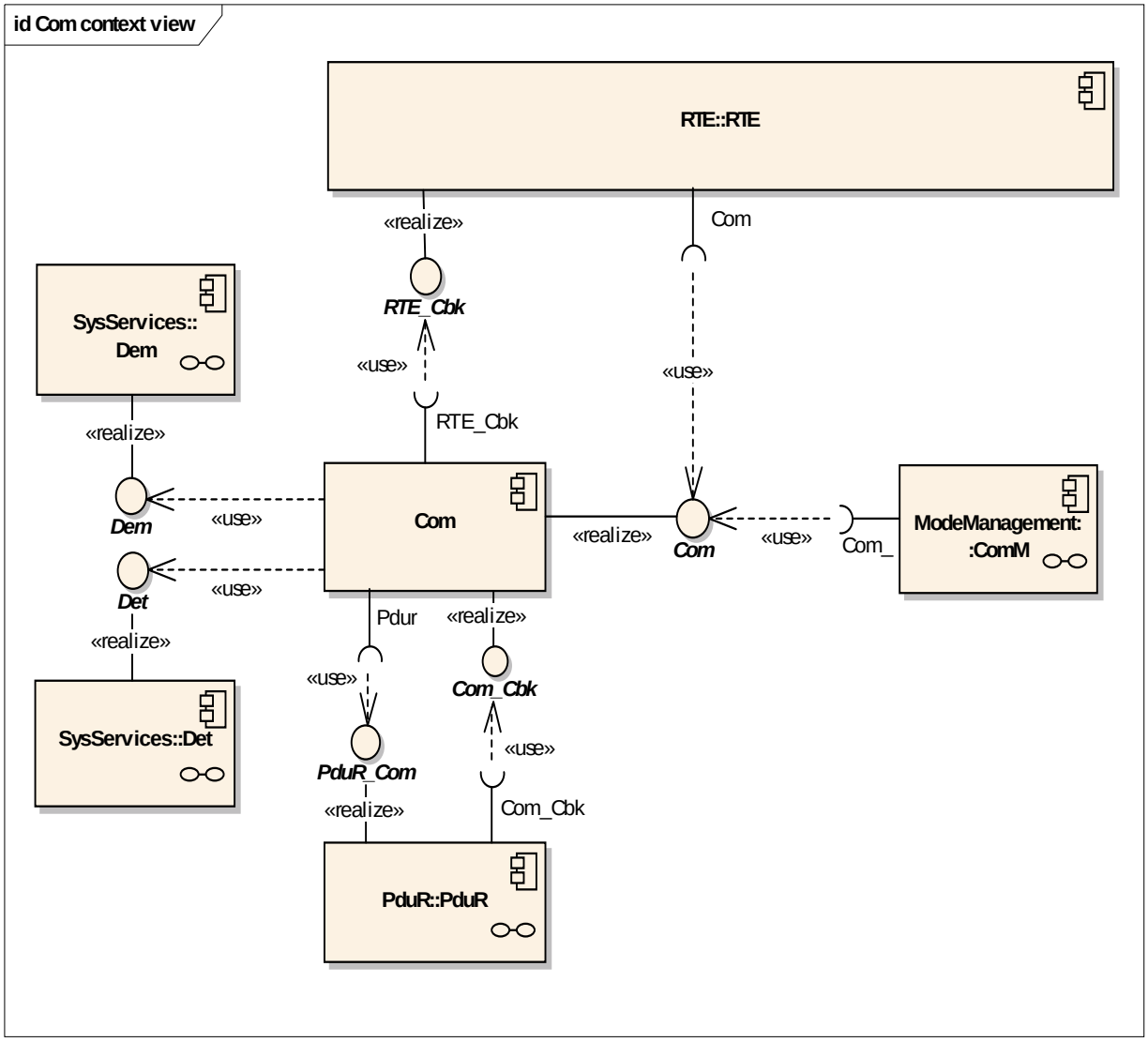

Figure 1: AUTOSAR COM context view

Note: RTE_Cbk denotes the interface provided by the RTE that is used by COM to notify about transmission acknowledgements and errors. This must not necessarily result in an RTE_Cbk.h file.

5.1 PDU router

The following summarizes the functionality AUTOSAR COM needs from the underlying layer PDU Router:

- Indication of incoming I-PDUs
- Sending interface for outgoing I-PDUs including the confirmation if an I-PDU has been send by the communication controller.
- Trigger interface to enable the PDU router to cause a transmission from AUTOSAR COM

A detailed description of the interface to the lower layer can be found in Chapter 7.6 and in Chapter 9.1. For further information see [10].

5.2 Runtime Environment (RTE)

RTE uses the capability to send and receive signals via AUTOSAR COM. In AUTOSAR the RTE is the higher layer above COM. For further information see [13].

5.3 COM Manager (ComM)

The COM Manager controls the start and stop of sending and receiving I-PDUs via AUTOSAR COM. For further information see [16].

5.4 File structure

5.4.1 Code file structure

COM430: The code file structure shall not be defined within this specification completely. At this point, it shall be pointed out that the code-file structure shall include the following files named:

- COM.c – module source file
- COM_PCcfg.c – pre-compile-time configurable constant parameters
- COM_Lcfg.c – link-time configurable parameters
- COM_PBcfg.c – post-build time configurable parameters

5.4.2 Header file structure

COM005: AUTOSAR COM shall offer a header file `Com.h` which includes all user relevant information and another header file `Com_Cbk.h` which declares the callback functions and the I-PDU callout prototype.

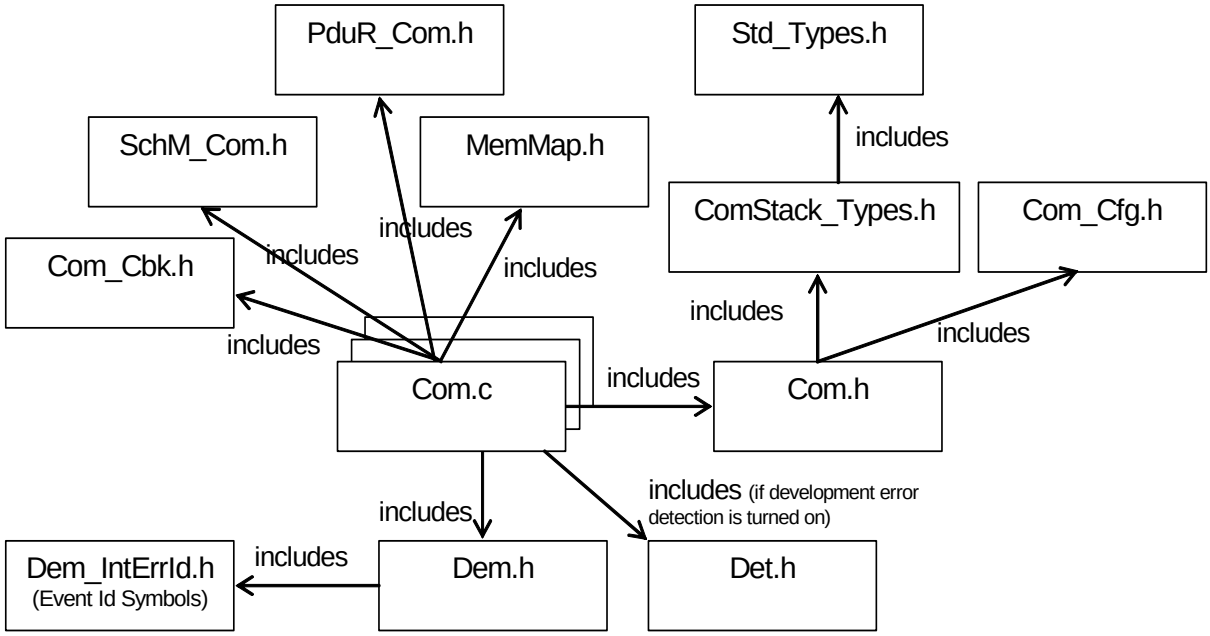

Figure 2: Include file structure

COM220: The include file structure regarding the specifics of the COM shall be constructed as shown in **Figure 2**.

COM431: The module shall include the Dem.h file. By this inclusion, the APIs to report errors as well as the required Event ID symbols are included. This specification defines the name of the Event ID symbols, which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event ID symbols and publishes the symbols in Dem_IntErrId.h.

6 Requirements traceability

Document: General Requirements on Basic Software Modules [3]

Requirement	*Satisfied by*
[BSW00344] Reference to link-time configuration	Chapter 10.2.1.2, COM432, COM430
[BSW00404] Reference to post build time configuration	Chapter 10.2, COM375, COM432
[BSW00405] Reference to multiple configuration sets	COM432
[BSW00345] Pre-compile-time configuration	Chapter 10.2.1.1, COM430
[BSW159] Tool-based configuration	not applicable (not in scope of this spec)
[BSW167] Static configuration checking	not applicable (not scope of this spec)
[BSW171] Configurability of optional functionality	not applicable (COM Module has no features switches for optional functionality)
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	not applicable (not in scope of this spec)
[BSW00380] Separate C-Files for configuration parameters	COM430
[BSW00419] Separate C-Files for pre-compile time configuration parameters	COM430
[BSW00381] Separate configuration header file for pre-compile time parameters	COM220
[BSW00412] Separate H-File for configuration parameters	COM220
[BSW00383] List dependencies of configuration files	not applicable (implementation specific)
[BSW00384] List dependencies to other modules	Chapter 5
[BSW00387] Specify the configuration class of callback function	Chapter 8.4
[BSW00388] Introduce containers	Chapter 10.2
[BSW00389] Containers shall have names	Chapter 10.2
[BSW00390] Parameter content shall be unique within the module	Chapter 10.2
[BSW00391] Parameter shall have unique names	Chapter 10.2
[BSW00392] Parameters shall have a type	Chapter 10.2
[BSW00393] Parameters shall have a range	Chapter 10.2
[BSW00394] Specify the scope of the parameters	Chapter 10.2
[BSW00395]	Chapter 10

Requirement	*Satisfied by*
List the required parameters (per parameter)	(scope and dependency fields)
[BSW00396] Configuration classes	Chapter 10.2
[BSW00397] Pre-compile-time parameters	Chapter 10.2
[BSW00398] Link-time parameters	Chapter 10.2
[BSW00399] Loadable Post-build time parameters	Chapter 10.2
[BSW00400] Selectable Post-build time parameters	Chapter 10.2
[BSW00402] Published information	Chapter 10.3
[BSW00375] Notification of wake-up reason	not applicable (not in scope of this spec)
[BSW101] Initialization interface	COM128, COM328, COM015, COM098, COM117, COM017, COM170, COM338, COM217, COM059, COM133COM432
[BSW00416] Sequence of Initialization	not applicable (not in scope of this spec)
[BSW00406] Check module initialization	COM433
[BSW00437] NoInit-Area in RAM	not applicable (not in scope of this spec)
[BSW168] Diagnostic interface	not applicable (not in scope of this spec)
[BSW00407] Function to read out published parameters	COM426
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	not applicable (not valid for AUTOSAR COM module)
[BSW00424] BSW main processing function task allocation	not applicable (implementation specific)
[BSW00425] Trigger conditions for schedulable objects	COM398, COM399, COM400, COM359
[BSW00426] Exclusive areas in BSW modules	not applicable (implementation specific)
[BSW00427] ISR description for BSW modules	not applicable (not valid for AUTOSAR COM module)
[BSW00428] Execution order dependencies of main processing functions	not applicable (not valid for AUTOSAR COM module)
[BSW00429] Restricted BSW OS functionality access	not applicable (AUTOSAR COM has no interface to OS)
[BSW00431] The BSW Scheduler module implements task bodies	not applicable (not in scope of this spec)
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	COM398, COM399, COM400, COM359, COM466
[BSW00433] Calling of main processing functions	not applicable (not in scope of this spec)
[BSW00434] The Schedule Module shall provide an API for exclusive areas	not applicable (not in scope of this spec)

Requirement	*Satisfied by*
[BSW00336] Shutdown interface	COM129, COM130
[BSW00337] Classification of errors	Chapter 7.10
[BSW00338] Detection and Reporting of development errors	COM024, COM028, COM442 Configuration: COM141
[BSW00369] Do not return development error codes via API	COM442, COM459, Chapter 8
[BSW00339] Reporting of production relevant error status	COM459, COM428
[BSW00417] Reporting of Error Events by Non-Basic Software	not applicable (not in scope of this spec)
[BSW00323] API parameter checking	COM024, COM028
[BSW004] Version check	COM026, COM337
[BSW00409] Header files for production code error IDs	COM431
[BSW00385] List possible error notifications	COM442, COM459
[BSW00386] Configuration for detecting an error	not applicable (not in scope of this spec)
[BSW161] Microcontroller abstraction	not applicable (not in scope of this spec)
[BSW162] ECU layout abstraction	not applicable (not in scope of this spec)
[BSW005] No hard coded horizontal interfaces within MCAL	not applicable
[BSW00415] User dependent include files	Chapter 5.4, COM005, COM220
[BSW164] Implementation of interrupt service routines	not applicable (not in scope of this spec)
[BSW00325] Runtime of interrupt service routines	not applicable (not in scope of this spec)
[BSW00326] Transition from ISRs to OS tasks	not applicable (not in scope of this spec)
[BSW00342] Usage of source code and object code	Chapter 10.2
[BSW00343] Specification and configuration of time	Chapter 10.2
[BSW160] Human-readable configuration data	Chapter 10.2
[BSW007] HIS MISRA C	API is MISRA conform other issues are implementation specific
[BSW00300] Module naming convention	COM005, COM220
[BSW00413] Accessing instances of BSW modules	not applicable (not in scope of this spec)
[BSW00347] Naming separation of different instances of BSW drivers	not applicable (not in scope of this spec)
[BSW00305] Self-defined data types naming convention	see Chapter 8.2
[BSW00307] Global variables naming convention	not applicable (implementation specific)

Requirement	*Satisfied by*
[BSW00310] API naming convention	Chapter 8.3 and 8.4
[BSW00373] Main processing function naming convention	Chapter 8.5
[BSW00327] Error values naming convention	COM442, COM459
[BSW00335] Status values naming convention	Chapter 8.2.1
[BSW00350] Development error detection keyword	COM028
[BSW00408] Configuration parameter naming convention	Chapter 10.2
[BSW00410] Compiler switches shall have defined values	not applicable (implementation specific)
[BSW00411] Get version info keyword	COM425
[BSW00346] Basic set of module files	COM005, COM220
[BSW158] Separation of configuration from implementation	COM005, COM220
[BSW00314] Separation of interrupt frames and service routines	not applicable (not in scope of this spec)
[BSW00370] Separation of callback interface from API	Chapter 8
[BSW00435] Module Header File Structure for the Basic Software Scheduler	COM220
[BSW00436] Module Header File Structure for the Basic Software Memory Mapping	COM220
[BSW00348] Standard type header	COM220
[BSW00353] Platform specific type header	not applicable (implementation specific)
[BSW00361] Compiler specific language extension header	not applicable (implementation specific)
[BSW00301] Limit imported information	not applicable (implementation specific)
[BSW00302] Limit exported information	not applicable (implementation specific)
[BSW00328] Avoid duplication of code	not applicable (implementation specific)
[BSW00312] Shared code shall be reentrant	COM320, COM321
[BSW006] Platform independency	not applicable (implementation specific)
[BSW00357] Standard API return type	Chapter 8
[BSW00377] Module specific API return types	Chapter 7.10, COM459
[BSW00304] AUTOSAR integer data types	COM220, (implementation specific)
[BSW00355] Do not redefine AUTOSAR integer data types	Chapter 7.10 and Chapter 8.2
[BSW00378] AUTOSAR boolean type	not applicable (implementation specific)
[BSW00306]	not applicable

Requirement	*Satisfied by*
Avoid direct use of compiler and platform specific keywords	(implementation specific)
[BSW00308] Definition of global data	not applicable (implementation specific)
[BSW00309] Global data with read-only constraint	not applicable (implementation specific)
[BSW00371] Do not pass function pointers via API	Chapter 8
[BSW00358] Return type of `init()` functions	COM432
[BSW00414] Parameter of init function	COM432
[BSW00376] Return type and parameters of main processing functions	Chapter 8.5
[BSW00359] Return type of callback functions	COM468, COM491
[BSW00360] Parameters of callback functions	COM468, COM491
[BSW00329] Avoidance of generic interfaces	Chapter 8
[BSW00330] Usage of macros / inline functions instead of functions	COM434
[BSW00331] Separation of error and status values	Chapter 8, COM194
[BSW009] Module User Documentation	not applicable (implementation specific)
[BSW00401] Documentation of multiple instances of configuration parameters	Chapter 10
[BSW172] Compatibility and documentation of scheduling strategy	see Chapter 8.5, COM298 further item are implementation specific
[BSW010] Memory resource documentation	not applicable (implementation specific)
[BSW00333] Documentation of callback function context	not applicable (not in scope of this spec)
[BSW00374] Module vendor identification	COM208
[BSW00379] Module identification	COM417
[BSW003] Version identification	COM426, COM425, COM424, COM026, COM337, COM141, COM186, COM208, COM417, COM418, COM419, COM420, COM421, COM422, COM423, COM438
[BSW00318] Format of module version numbers	COM026
[BSW00321] Enumeration of module version numbers	not applicable (not in scope of this spec)
[BSW00341] Microcontroller compatibility documentation	not applicable (not in scope of this spec)
[BSW00334] Provision of XML file	not applicable (not in scope of this spec)

Document: Requirements on Communication [7]

Requirement	***Satisfied by***
[BSW02037] AUTOSAR COM shall be based on the functionality and APIs of OSEK COM 3.0.3	COM010, COM011, COM012, COM013, COM396
[BSW02078] Support of endianness conversion	COM007, COM221, COM352, COM472, COM473
[BSW02086] Support of Sign-Extension for received signals	COM008, COM353, COM352
[BSW02042] Initialization of not used areas of an I-PDU	COM015
[BSW02040] AUTOSAR COM Configuration Language	COM006, Chapter 10
[BSW177] Configuration of communication parameters	Chapter 10.2, COM490
[BSW02067] Rules for checking the consistency of configuration input	COM101, COM102, COM105, COM216, COM319, COM365, COM373, COM384, COM385, COM386, COM389, COM310, COM401, COM402, COM443, COM474, COM477, COM482, COM485, COM489
[BSW02046] Configuration of signal notification	COM298, COM300, COM301, COM393
[BSW02089] Timeout indication mechanism on receiver-side	[17] COM292, COM290, COM291, COM393 Configuration: COM263, COM264, COM333
[BSW02088] Value substitution in case of a signal timeout	COM393
[BSW02083] Transmission Modes	COM329, COM330, COM135, COM276, COM305, COM306COM392, COM277, COM278, COM285, COM307, COM308, COM467, COM478 Configuration: COM351, COM178, COM137, COM180, COM281, COM282
[BSW02082] Two different Transmission Modes	COM032, COM239, COM244, COM238 Configuration: COM454, COM455, COM465
[BSW02084] Transmission Mode Selection	COM274, COM283, COM241, COM245, COM284, COM255, COM032
[BSW02080] Re-Triggering of repetitions of I-PDUs	COM279
[BSW02077] Signal invalidation mechanism on sender-side	COM097, COM099, COM286 API: COM203, COM288 Configuration: COM338, COM314, COM315, COM316, COM391,
[BSW02079] Signal invalidation mechanism on receiver-side	COM100, COM287, COM323
[BSW02087] Substitution of invalid value by configurable data value	COM100, COM488, COM412, COM463, COM470

Requirement	*Satisfied by*
[BSW218] Separate Start/Stop AUTOSAR COM for separate groups of I-PDUs	COM085, COM114, COM222, COM223, COM228, COM229, COM334, COM090, COM115, COM311, COM187, COM444, COM476, COM479 API: COM190, COM191 Configuration: COM341, COM126, COM184, COM185
[BSW192] Disable Reception Deadline Monitoring	COM092, COM225 API: COM192,
[BSW02081] Re-enable Reception Deadline Monitoring	COM095, COM224, COM486 API: COM193
[BSW02041] Atomic transfer of complex data types	COM042, COM043, COM047, COM049, COM050, COM051, COM052, COM053, COM327, COM326, COM461, COM462, COM463, COM464 API: COM199, COM200, COM201, COM202 Configuration: COM345, COM044, COM149, COM349, COM350, COM152, COM415, COM416, COM441, COM452, COM453
[BSW02043] Indication service Com_RxIndication	API: COM123
[BSW02044] Confirmation service Com_TxConfirmation	COM260 API: COM124
[BSW02045] Function Com_TriggerTransmit	API: COM001, COM260, COM475
[BSW02030] Identify if a signal is updated by the sender	COM054, COM055, COM116, COM056, COM057, COM059, COM061, COM062, COM067, COM076, COM324, COM117, COM310 Configuration: COM257, COM258, COM429
[BSW02058] Deadline monitoring of Receiving Updated Signals	COM292, COM290, COM291, COM117, COM393

Document: OSEK/ VDX Communication Version 3.0.3 [17]

Requirement	*Satisfied by*
Filtering (Section 2.2.2 in [17])	COM325, COM380, COM272, COM273, COM132, COM230, COM302, COM303, COM231, COM439, COM480 Configuration: COM339, COM235, COM312, COM313, COM146, COM147, COM317, COM318
Reception Deadline Monitoring (Section 2.5.1 in [17])	COM292, COM290, COM291
Transmission Deadline Monitoring (Section 2.5.2 in [17])	COM304, COM445, COM458, COM481
I-PDU Callout (Section 2.9.3.2, Appendix C in [17])	COM381, COM346, COM347, COM395, COM388 API: COM348
OSEK APIs	COM197, COM198, COM469 Configuration: COM340, COM174, COM175, COM176, COM017, COM181, COM119, COM387, COM206, COM233, COM234, COM342, COM156, COM157, COM158, COM259, COM160COM183, COM263, COM264, COM333, COM161, COM267, COM268, COM356, COM344, COM163, COM165, COM166, COM232, COM170, COM127, COM045, COM167, COM169, COM275, COM409COM410, COM411, COM437, COM440, COM446, COM447, COM448, COM449, COM450, COM451, COM456, COM457, COM471
Notifications	API: COM123, COM124 Configuration: COM343, COM247, COM248, COM246,

Document: Requirements on Gateway [9]

Requirement	*Satisfied by*
Interface between AUTOSAR COM and the lower layer (PDU-Router)	COM138
[BSW06002] Updateable Configuration	COM373, COM357, COM361
[BSW06097] Configuration identification	COM374, COM375, COM394, COM487
[BSW06003] Static Routing Rules	COM376 Configuration: COM355, COM358, COM382
[BSW06055] Signal Based Gateway	COM377
[BSW06056] Gateway of Signal Groups	COM361, COM383

Requirement	*Satisfied by*
[BSW06061] Routing operation on Signals	COM371, COM360, COM361, COM362
[BSW06098] Signal Gateway Error Handling at signal routing	COM024, COM442, COM459
[BSW06099] Signal Gateway Error Handling at signal routing	COM024, COM442, COM459
[BSW06077] Routing of multiple signals of the same PDU	COM371
[BSW06089] Timeout handling	COM381
[BSW06089] Routing of invalid value	COM377
[BSW06064] Signal gateway scalability	COM370

7 Functional specification

7.1 Introduction

7.1.1 Signal values

The signals sent by AUTOSAR COM respectively received by AUTOSAR COM could have the values defined in Table 1.

Signal value	*Remark*
init value	See Chapter 7.4.1.4 for details.
invalid value	See Chapter 7.4.3.6 for details.
<value>	This is the normal case: A valid value after initialization phase which is sent by AUTOSAR COM respectively received by AUTOSAR COM.

Table 1: Possible signal values

7.2 General functionality

7.2.1 OSEK-COM

OSEK COM 3.0.3 is the functional basis of AUTOSAR COM.

COM010: AUTOSAR COM shall implement all the functionality and all the APIs of OSEK/ VDX Communication Version 3.0.3 [17] except the features and APIs mentioned in COM013.

COM011: If this specification defines functionality in a different way compared to definitions in [17], the definitions made in this AUTOSAR COM specification shall be used.

COM012: AUTOSAR COM shall, in addition, implement all those features, that are defined in this specification and that are not part of [17].

COM013: AUTOSAR COM needs *not* to implement the following features of [17]. If they are implemented in a specific AUTOSAR COM configuration the configuration shall by default disable them. This also applies for all other additional features a specific implementation may offer.

OSEK-COM feature	*Rationale*	*related OSEK COM API*
mapping of a received network message (within an I-PDU) to more than one message data objects (1:n splitting mechanism)	not required, done by the RTE (see [13])	none
mapping of an internal message to more than one message data objects (1:n splitting mechanism)	not required, done by the RTE (see [13])	none

OSEK-COM feature	*Rationale*	*related OSEK COM API*
mapping an only locally send message to both an external send message object and an internal receive message object (1:n splitting mechanism)	not required, done by the RTE (see [13])	none
M:1 sending; mapping of messages from multiple senders to one and the same message object	not required, ensured by RTE (see [13])	SendMessage
queued messages	not required, done by the RTE (see [13])	GetMessageStatus
zero size messages	it is possible to set up communication without them functionality is partly covered by COM_TriggerTransmit()	SendZeroMessage
dynamic size messages	were introduced for diagnostics, no use case in AUTOSAR	SendDynamicMessage, ReceiveDynamic-Message
notification mechanisms TASK, FLAG and EVENT	not required, done by the RTE (see [13])	none
overlapping messages in an I-PDU	no use case, dangerous concept	none
usage of OIL	not the OSEK OIL shall be used to configure the AUTOSAR COM	none
Application modes	not needed	GetApplicationMode
Start-up behavior	replaced by Com_Init(), Com_DeInit(), Com_IpduGroupStart(), Com_IpduGroupStop()	StartCOM, StopCOM, StartCOMExtensions, InitMessage
start and stop of periodic messages	no use case, is realized by I-PDU group mechanism	StartPeriodic, StopPeriodic
Reentrancy	Not all of the AUTOSAR API calls are reentrant. See Chapter 8.3.	See Chapter 8.3.
Interface to OSEK indirect NM	not needed	I_MessageTransfer, I_MessageTimeOut
Sender side filtering	no use case, the filter conditions are still used in the selection of the Transmission Mode but there is no signal filtering	none
Network-order Message Callout CPU-order Message Callout	Only I-PDU callouts with a defined AUTOSAR interface are supported by AUTOSAR COM. This is to avoid proprietary solutions.	none
Error hook routine	AUTOSAR COM will use a direct interface to DEM/DET instead of using the OSEK COM error hook	COMErrorHook COMError_Name1_Name2 macros COMErrorGetServiceId
Interface for callback routines	The signatures for the used callback function in AUTOSAR COM will be explicitly defined within AUTOSAR COM.	COMCallback

Table 2: Excluded OSEK COM features in AUTOSAR COM

7.2.2 Endianness conversion and sign extension

COM007: To support the required AUTOSAR data types (signed- and unsigned integer, ASCII, enum, opaque bitfield) the AUTOSAR COM layer shall provide support for endianness conversion of all integer types. Other data types (ASCII, enum, opaque[1]) are either treated as signed or unsigned integer or nothing has to be done, i.e. their contents is not interpreted by the AUTOSAR COM layer.

The supported data types of AUTOSAR COM are:

- boolean
- uint8
- uint16
- uint32
- sint8
- sint16
- sint32
- uint8[n]

COM472: Opaque data shall always be of uint8[n] and shall always be mapped to an n-bytes sized network signal.

Note: For opaque data (see [13]) endianness conversion shall be configured to ***Opaque*** (see COM157).

COM473: If the endianness conversion is configured to *Opaque* (see COM157) the parameter COM_NETWORK_SIGNAL_BIT_POSITION (see COM259) shall define the bit0 of the first byte like in little endian byte order.

Remark: [17] Chapter 2.4 defines the endianness conversion for unsigned data types. This endianness conversion shall be extended to signed data types.
The associated configurations can be found in the configuration chapter. See also COM127 and COM157.

COM008: To map negative values of signed data correctly, the received data shall be extended to the size of the receiver (sign extension). The platform specific representation of signed data shall be taken into account.

Example: A 10-Bit signed network signal is received and shall be copied by `Com_ReceiveSignal()` to a 16-Bit signed integer variable. If $(-3)_{decimal}$ is received the received 10-Bit network signal has a value of `1111111101b`. While copying it to the 16-Bit integer variable the value has to be extended to `1111111111111101b`.

[1] This Data type represents an array of exactly numberOfBits bits. It is called *opaque* because this array of bits should be transported "as is" by the AUTOSAR communication stack.

COM353: On sender side there shall be no sign extensions. (See COM389).

COM221: Endianness conversion shall take place before the I-PDU callout on the sender side. (For an overview see Chapter 7.12).

COM352: Sign extensions and endianness conversion shall take place before configuration filtering and notification detection on receiver side

7.2.3 Filtering

COM272: Each filtering condition shall either evaluate to true or false. Filtering out signals shall only take place at receiver side. On sender side the mechanisms are still used for Transmission Mode Conditions but no signal filtering shall take place.

Note: For Transmission Mode Selection see Chapters 7.4.3.4 and 7.4.3.3.

COM480: AUTOSAR COM shall only provide the following filter types which are defined in [17]

- F_Always
- F_Never
- F_MaskedNewEqualsX
- F_MaskedNewDiffersX
- F_MaskedNewDiffersMaskedOld
- F_NewIsWithin
- F_NewIsOutside
- F_OneEveryN

Note: Some filters defined in [17] are removed from AUTOSAR COM to reduce complexity. The removed filters were either obsolete or special cases of other filters. For example the filter F_NewIsDifferent is a special case of F_MaskedNewDiffersMaskedOld with a fully set mask.

COM325: All filter mechanisms shall be supported for all data types listed in COM007 considering the exceptions defined in COM380 and COM439.

COM380: For the type uint8[n] no filter mechanisms shall be supported.

COM439: For the type boolean only the following filters shall be supported:

- F_Always
- F_Never
- F_MaskedNewEqualsX
- F_MaskedNewDiffersX
- F_MaskedNewDiffersMaskedOld
- F_OneEveryN

COM273: If a signal is filtered out on receiver side (i.e. filter condition evaluates to FALSE), it shall be discarded and not be further processed. See also COM303.

COM327: Filtering out of signals as specified in COM273 shall not be applied to signals contained in signal groups.

Note: Conditions for TMS may be applied to signals contained in a signal group, see COM326.

COM132: The filtering mechanisms, defined in [17] and in Chapter 7.4.3.3 shall also apply for signed data types.

In the case a filter is evaluated before a signal has been written to by a Send-API, there needs to be a way to determine the filter state of this signal. Some of the filters require a *new_value* to evaluate the filter. However, this is only available after the signal has been updated using a Send-API. So it is necessary to define the value used by the filter for *new_value* in the period before the first send takes place.

COM230: The *old_value* of the filtering mechanisms is set during start-up to the init-value, see [17]. Up to the point in time where the application has not updated the value, for the *new_value* also the init-value shall be used.

The next two requirements shall clarify the definitions of [17] according to the update of the *old_value* of filters.

COM302: If a filter is evaluated to TRUE (value is not filtered out) then the value is placed into *old_value* (as defined in [17]).

COM303: When a value is being filtered, if the filter does not allow the passage of the value (i.e. the filter evaluates to FALSE) then the value is not placed into *old_value* (as defined in [17]).

COM231: In the case of the filter F_OneEveryN,

- OCCURRENCE shall be set to zero by `Com_IpduGroupStart`
- FILTER shall be set TRUE when OCCURRENCE == OFFSET
- OCCURRENCE shall be incremented after filter processing
- When OCCURRENCE == PERIOD, OCCURRENCE shall be set to zero.

For definition of OCCURRENCE, FILTER, OFFSET and PERIOD see [17].

This definition of the filter F_OneEveryN has the effect that the signal is passed by the filter (i.e. the filter returns TRUE) once every PERIOD call of the filter. If the OFFSET parameter is zero then the first time the filter is used the signal is allowed to pass (i.e. filter returns TRUE). If the OFFSET is greater than zero then more than one message must pass through the filter before it returns TRUE.

This definition exists to clarify the description of the F_OneEveryN filter in [17].

The associated configurations can be found in the configuration chapter at COM147.

7.2.4 Signal Gateway

AUTOSAR COM provides a Signal Gateway for forwarding signals and signal groups in a 1:n manner.

Signals and signal groups to be routed by the Signal Gateway are identified and configured by unique static names. The Signal Gateway determines the destination of a signal or of a signal group by using its name and a configuration table.

COM370: Furthermore the Signal Gateway should scale down to no size if no signal routing functionality is needed.

COM371: The Signal Gateway is placed as shown in Figure 3, Figure 13, Figure 14 and Figure 15.

7.3 Configuration

See Chapter 10.

7.4 Normal operation

7.4.1 Start-up behavior

This chapter describes the actions which shall be performed during `Com_Init()`.

COM217: The I-PDU init value (Chapter 7.4.1.3) and the signal / update bit init value (Chapter 7.4.1.4) together result in one init value for each I-PDU. During `Com_Init()` this init value shall be used to initialize the I-PDU.

7.4.1.1 Preconditions

Note: The C initialization code (also known as *start-up code*) which initializes global and static variables with the initial values shall be executed before any call of an AUTOSAR COM service.

7.4.1.2 Initialization

COM128: All internal data that is not yet initialized by the *start-up code* e.g. C-structs shall be initialized by `Com_Init()` (see COM432).

COM328: The AUTOSAR COM initialization routine shall enable ECU internal communication but not Inter-ECU communication. Inter-ECU communication requires a separate start (see Chapter 7.4.5 for details).

The API function is defined by COM432.

Note: This initialization chapter is not complete. Details about initialization of some COM features are described within the different feature chapters.

7.4.1.3 Initialization of not used areas of an I-PDU

COM015: AUTOSAR COM shall fill not used areas within an I-PDU with a value configurable at configuration-time (e.g. 0xFF).

The associated configurations can be found in the configuration chapter, see COM017.

7.4.1.4 Initialization of signals and Update-bits

COM098: AUTOSAR COM shall initialize all signals on sender and receiver side with the configured init values which shall be the same value as used for the initialization of the signal in the related I-PDU (see COM217). For configuration see COM170 and COM338 in the configuration chapter.

The signal init values may be identical to the AUTOSAR COM *Data Invalid Values* (see COM316). These may be different for each signal.

COM117: AUTOSAR COM shall clear all Update-bits during initialization. (See also COM059).

7.4.1.5 Initialization of I-PDU groups

COM444: By default all I-PDU groups shall be in the state ‘stopped’ and they shall not be started automatically while AUTOSAR COM initialization.

7.4.2 Shutdown behavior

7.4.2.1 De-initialization

COM129: AUTOSAR COM shall provide a functionality to de-initialize the layer. This means that after de-initialization of the layer no communication via AUTOSAR COM is possible and all started I-PDU groups are stopped.

The API function is defined by COM130.

7.4.3 Communication modes

This chapter defines the signal flow in AUTOSAR COM and how the different Transmission Modes in AUTOSAR COM are defined. Chapter 7.4.3.6 shows exemplary communication use cases that can be solved with AUTOSAR COM. Chapter 7.4.3.3 defines a mechanism to switch between two Transmission Modes for one I-PDU. The replication of signals is defined in Chapter 7.4.3.5.

7.4.3.1 Signal Transfer Property and I-PDU Transmission Mode

[17] distinguishes between

Transfer Properties (applies for signals)
Transmission Modes (applies for I-PDUs).

The following two tables are derived from [17]. ***The AUTOSAR extensions are marked with bold and configuration letters.***

TRANSFER PROPERTY ***(per signal)***	***Description***
Triggered	**COM329:** The Triggered Transfer Property causes immediate transmission of the I-PDU, except if Transmission Mode "Periodic" or ***Transmission Mode "None"*** is defined for the I-PDU (see tabular below).
Pending	The "Pending" Transfer Property does not cause transmission of an I-PDU.

Table 3: Transfer Properties defined for signals

Note: A pending signal associated with an I-PDU is transmitted if either a signal with Triggered Transfer Property within the same I-PDU is sent or the I-PDU is sent because of a timer (Periodic / Mixed Transmission Mode).

Transmission Mode ***(per I-PDU)***	***Description***
Direct/N-Times	**COM330**: Transmission of an I-PDU with Direct Transmission Mode is caused by the transfer of any signal assigned to the I-PDU with the "Triggered" Transfer Property. The transfer of the signal to AUTOSAR COM is immediately followed by ***n transmissions*** (n = 1 … m, m <= 255) on the underlying layer (associated requirements see 7.4.3.5).
Periodic	In "Periodic" Transmission Mode, AUTOSAR COM issues periodic transmission requests for an I-PDU to the underlying layer.
Mixed	Mixed Transmission Mode is a combination of the "Direct/N-Times" and the "periodic" Transmission Modes.
none	**COM135**: ***In Transmission Mode "none" no transmission is initiated by AUTOSAR COM. It shall only be possibly to request I-PDUs with Com_TriggerTransmit() (see COM001).*** ***The configuration is defined by COM137.***

Table 4: Transmission Modes defined for I-PDUs

On the one hand the timing of bus messages can be controlled by send requests of the upper layer in combination with the Transmission Mode and Transfer Property as described above. On the other hand it can be controlled by the lower layer (esp. FlexRay and LIN) with the service `Com_TriggerTransmit()`. In this case the lower layer only requests I-PDUs that have to be provided by COM.

COM260: It shall be possible to use `Com_TriggerTransmit()` to request the transmission of any I-PDU with any Transmission Mode.

Note: This allows LIN and FlexRay to use all the available Transmission Modes, particularly for sporadic communication. This mechanism is also used by the NM to send user data.

7.4.3.2 Link to the VFB specification

Note: This chapter is just an illustration how the transfer mode relates to the VFB specification and links to a non normative part of the VFB specification.

Because, from the point of view of the AUTOSAR SW Component, it is not known at implementation time, which communication media is used, all bus specific replications of send requests by a SW component to underlying layers as well as all bus specific timing behavior must be done either by AUTOSAR COM or by the appropriate bus interfaces and drivers.

AUTOSAR COM implements the replication of transmission requests and the bus specific timing behavior by a combination of Transfer Properties and Transmission Modes, which is shown in the table below. The entries in the table correspond to the VFB's send modes:

Transfer Property (horizontal) ***Transmission Mode (vertical)***	***Triggered***	***Pending***
Direct/N-Times	N-Times	--
Periodic	--	Cyclic
Mixed	N-Times	Cyclic
None	--	--

Table 5: Mapping of Transfer Property and Transmission Mode (I-PDU) to send modes defined in the AUTOSAR Specification of the Virtual Functional Bus [6]

7.4.3.3 Selection of the Transmission Mode for one specific I-PDU

Signals are carried by I-PDUs. Because an I-PDU can contain more than one signal, in the following, a method is defined to derive the I-PDU's Transmission Mode from the state of the signals that are contained in one specific I-PDU.
AUTOSAR COM allows configuring statically two different Transmission Modes for each I-PDU (see COM032). The Transmission Mode of an I-PDU that is valid at a specific point in time is selected using only the values of the signals that are mapped to this I-PDU.
The signals of one I-PDU that contribute to the selection of one of the two Transmission Modes as well as the conditions used for the selection of the Transmission Mode are configured statically.

COM326: For the selection of the Transmission Mode the signals of a signal group shall be treated as if they do not belong to a signal group. Therefore each signal belonging to a signal group shall contribute, depending on its configuration, to the evaluation of the Transmission Mode.

The following definitions give an overview about terms used in the following specification articles.

Definitions	
s_i	Signal in AUTOSAR COM
$IPDU_k$	I-PDU in AUTOSAR COM
$C(s_i , IPDU_k)$	Transmission Mode Condition, condition attached to a signal (s_i) within an I-PDU ($IPDU_k$)
S_k	The set of all signals (s_i) within one I-PDU ($IPDU_k$)
M_k	The set of all signals (s_i) within one I-PDU ($IPDU_k$) which contribute according to their configuration to the selection of the Transmission Mode. M_k has to be a subset of S_k.
TMS_k	Transmission Mode Selector, is calculated from the evaluation of all $C(s_i , IPDU_k)$ for all s_i in M_k

COM274: To each signal (s_i) mapped to a specific I-PDU ($IPDU_k$) a condition $C(s_i , IPDU_k)$ shall be associated (see COM275).

COM283: The conditions $C(s_i , IPDU_k)$ available shall be the same as provided for the filter mechanism (see COM272 and COM273).
The definitions made in COM230 and COM231 shall also be applied for the evaluation of the TMS.

Definition (see also [17]):
For each I-PDU ($IPDU_k$) a Transmission Mode Selector (TMS_k) is defined. TMS_k is calculated by evaluating the conditions $C(s_i , IPDU_k)$ for a configurable subset of signals M_k (see COM275) of the set of all signals S_k mapped to the $IPDU_k$.

TMS_k is defined to be true, if at least one $C(s_i , IPDU_k)$ in M_k evaluates to true and is defined to be false, if all $C(s_i , IPDU_k)$ in M_k evaluate to false.

$$TMS_k = \begin{cases} \text{TRUE, if at least one } C(s_i, IPDU_k) \equiv \text{TRUE for all } s_i \in M_k \\ \text{FALSE, if } C(s_i, IPDU_k) \equiv \text{FALSE for all } s_i \in M_k \end{cases}$$

COM241: AUTOSAR COM shall be able to define and calculate a TMS for each I-PDU as defined in the definition given above.

COM245: The TMS of each I-PDU containing a specific signal shall be re-calculated within a call of `Com_SendSignal()` respective `Com_SendSignalGroup()` by upper layers for that specific signal.

COM284: AUTOSAR COM must provide the mechanisms to specify which signals shall contribute to the evaluation of TMS, i.e. to define the set of signals M_k that contribute to the evaluation of the TMS_k .

For the configuration see COM275.

COM255: If no signal within an I-PDU is configured to contribute to the calculation of the TMS, the TMS shall be set to TRUE.

Comment: Note that a signal with C(s_i , $IPDU_k$) F_ALWAYS – if it contributes to the selection of the Transmission Mode – will automatically always set the TMS of the respective I-PDU to TRUE. Therefore, care must be taken when defining the signals that contribute to the TMS.

COM032: On sender-side two independent Transmission Modes (see also [17]) shall be configured for each I-PDU. One of those Transmission Modes shall be valid if the TMS evaluates to TRUE, the other Transmission Mode shall be valid, if the TMS evaluates to FALSE.

The Transmission Modes, which can be configured separately for each evaluation result, are defined in the configuration chapter, see COM233 and COM234.

COM238: In each of the two TMS states, the rules for combination of Transfer Properties of signals and Transmission Modes of I-PDUs shall apply as defined in [17], Section 2.3.

COM239: When the TMS state changes, the now valid Transmission Mode shall immediately be used. That means, first, the mode change shall be done and after that the appropriate requests to the underlying layers, resulting from the request causing the mode change, shall be executed.

COM244: If a change of the TMS causes a change of the Transmission Mode for one I-PDU, the timer for the cycle time of the Periodic and the Mixed Transmission Mode shall be restarted.

Remark: In case the TMS evaluates to FALSE, it shall be possible to configure the Transmission Mode in that way that no transmission takes place (Transmission Mode None). This prevents the I-PDU to be transmitted and shall be the default. For the configuration see COM234.

COM478: An I-PDU shall be sent out at most once while one call to `Com_MainFunctionTx()`.

7.4.3.4 Signal flow and Transmission Mode Selection

After a send request from an upper layer for a specific signal the signal is written to the appropriate I-PDU buffer as defined by configuration and the selection of the Transmission Mode of the I-PDUs is done according to Chapter 7.4.3.3.

Figure 3 shows the signal flow:

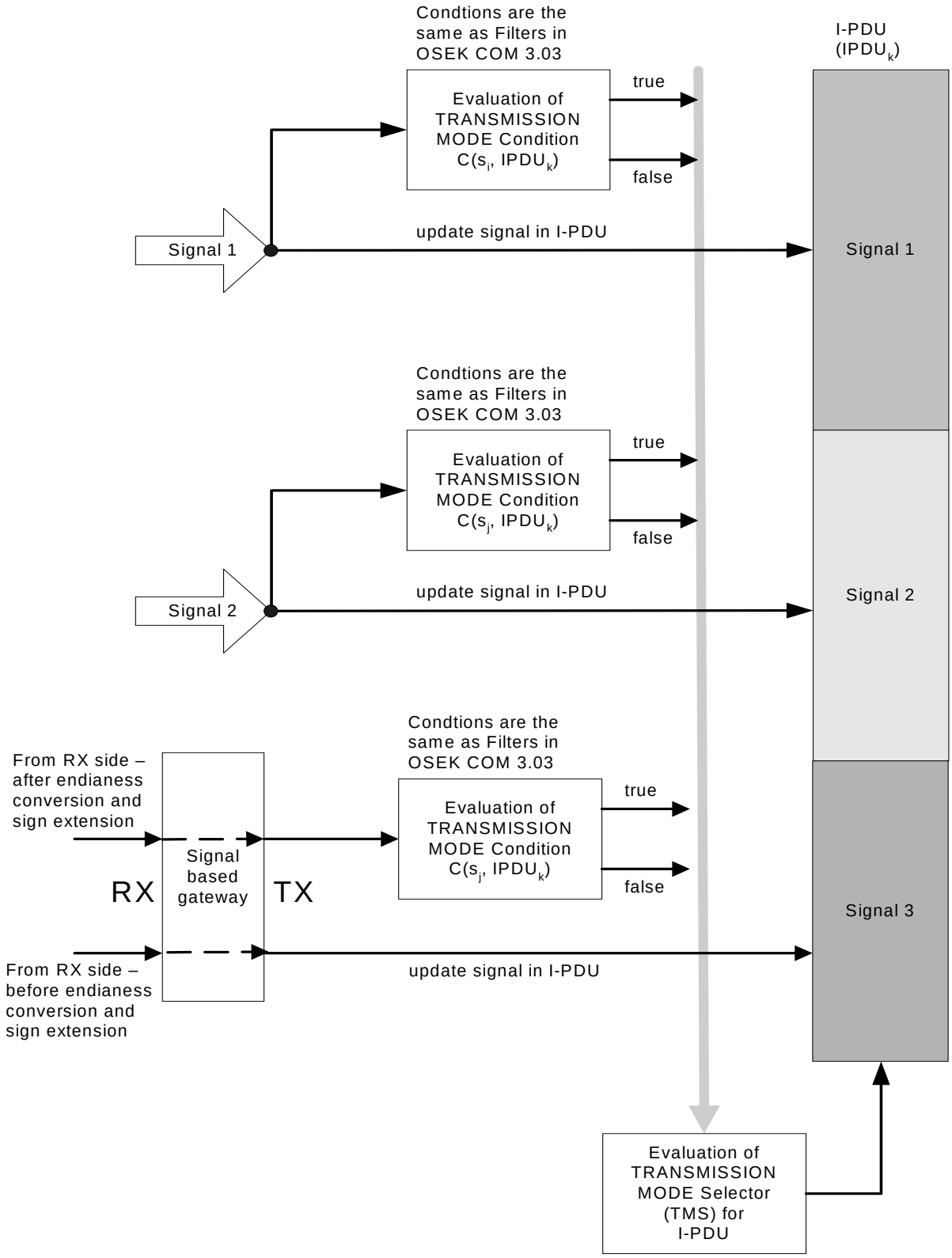

Figure 3: Logical signal flow in AUTOSAR COM shown for two signals (Signal1 and Signal2) that are mapped to one I-PDU (IPDU$_k$)

7.4.3.5 Replication of Signal Transmission Requests

COM276: In the Direct/N-Times and Mixed Transmission Modes AUTOSAR COM shall be able to send n transmission requests (n = 0, 1 … m, m <= 255) to the lower layer for each send request by an upper layer. See also the matching configuration requirement COM281.

COM467: If Direct/N-Times or Mixed Transmission Mode with n == 0 is used this shall result in only one send request without waiting for any confirmation (original OSEK behavior for Direct Transmission Mode).

COM279: If a new send request is received from an upper layer while sending n transmissions belonging together (e.g. after the 3rd of 5 repetitions, see COM276) AUTOSAR COM shall cancel the outstanding transmission repetitions and start processing the new request immediately (see Figure 4).

COM305: The confirmation behavior to the upper layer in the Direct/N-Times Transmission Mode with n > 0 shall be according to the following definition:

1) When an I-PDU is sent by the application with Direct/N-Times Transmission Mode, n is put into a counter in COM.
2) COM sends to the underlying layer an N-Times I-PDU periodically as long as the counter is non-zero.
3) Whenever a TX confirmation is received and the counter is greater than 0, the counter is decremented. When the counter is 0, transmission confirmations for that I PDU are ignored.
4) When the counter reaches 0 the transmission confirmation is send to the upper layer and Transmission Deadline Monitoring is cancelled (if configured). See COM392 and Chapter 7.4.6.2.

Note: The definition in COM305 shall not define a concrete implementation. But every implementation shall implement the confirmation behavior according to the above definition.

Note: This solution allows the violation of the period in certain extreme circumstances when the confirmations arrive late in the period.
This solution requires that CAN does not have a queue for these L-PDUs.
There is a race condition in the interaction between the CAN driver, interface and hardware that may cause an extra 1 transmission to occur in certain unlikely circumstances.

Note: If the underlying layer returns E_NOT_OK while an N-Times Transmission is in progress this error notification shall be ignored (as defined in COM428). As COM305 specifies only confirmed transmissions are counted for the N-Times Mode, erroneous send request can safely be ignored.

COM392: If a Transmission Deadline Monitoring timeout occurs before N-Times Transmission is complete, the N-Times Transmission shall be abandoned.

Note: The minimum delay time shall always be taken into account as defined in [17].

Note: To avoid bursts in start-up a time offset can be configured per I-PDU. For details see COM180.

COM469: The minimum delay time counter is started whenever a confirmation for that I-PDU is received.

COM277: The number of repetitions (n) shall be configurable (see COM281).

COM278: The time between two repetitions shall be configurable (see COM282).

COM285: If the Transmission Mode change leads to the start of the Mixed Transmission Mode, at the beginning of the Mixed Transmission Mode at least n transmission requests (for definition of n see COM276) to lower layers shall be executed (see Figure 11).

COM310: It shall not be possible to use/configure Update-bits for signals that are transmitted within I-PDUs having the Direct/N-Times Transmission Mode with n>=1. (I.e. only n=0 is allowed).

Note: There is no common understanding how Update-bits and N-Times Transmission can be combined. It is unclear when Update-bits shall be set and cleared respecting the replication. Therefore this is currently forbidden.

7.4.3.6 Use Cases for communication modes

This chapter shows Use Cases which have to be realizable by the Transmission Modes and Transfer Properties specified in AUTOSAR COM.

Note: For a more detailed discussion of the following use cases see Appendix A.

Use-Case diagram legend	
t_c, t_{c1}, t_{c2}	Cycle times
t_d	Cycle time of N-Times send signals
$t_{r, min}$	Minimum SW reaction time of COM-Layer, e.g. due to internal cycle time
V	Value: x stands for an arbitrary value / value range, a…w for specific values / value ranges, defined by the user, with a <> b, *range a* is disjoint from *range b*.
	Request from upper layers (RTE) to the COM-Layer
	Request from COM-Layer to lower layers (PDU-Router)
	Potential but skipped request from COM-Layer to lower layers (e.g. because of a new send request by upper layers or delayed due to minimum delay time)
dt	Minimum distance between two requests to lower layers (minimum delay time), dt can be set per I-PDU.
w/o TMS switch	Without switching of the TMS (see 7.4.3.3) from *true* to *false* or vice versa.
w TMS switch	With switching of the TMS (see 7.4.3.3) from *true* to *false* or vice versa (from TM 1 to TM 2) One TM is named before the “+” and one behind in the description.

Table 6: Legend for Use Case diagrams.

Use Case diagrams:

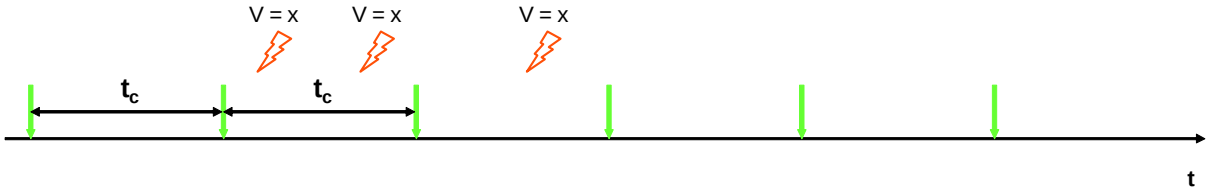

Figure 4: Use Case 1, TM Periodic (without TMS switch)

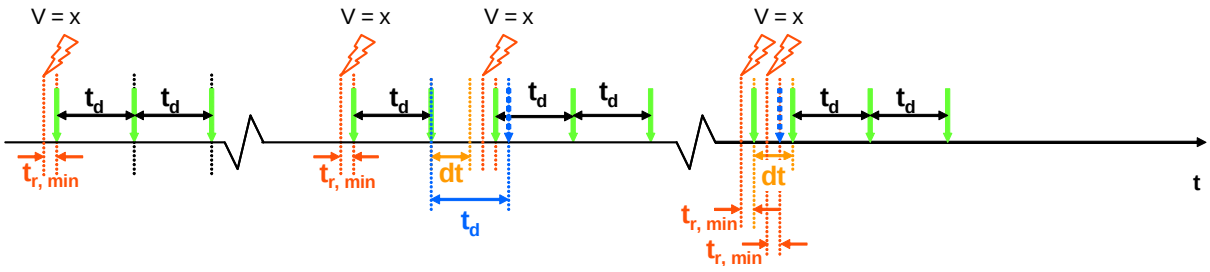

Figure 5: Use Case 2, TM Direct/N-Times, here n = 3 (without TMS switch)

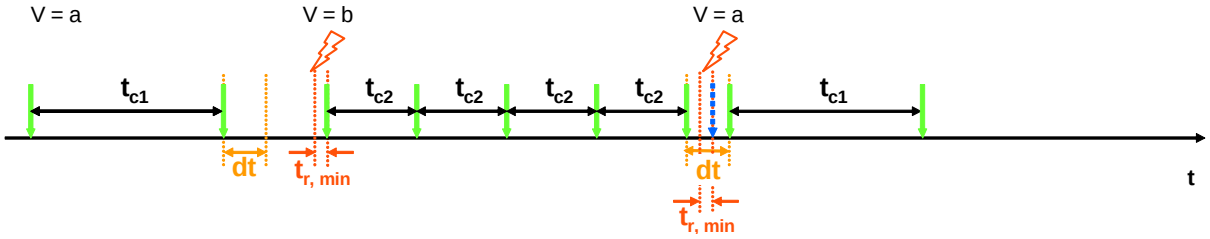

Figure 6: Use Case 3, TM Periodic + Periodic (with TMS switch)

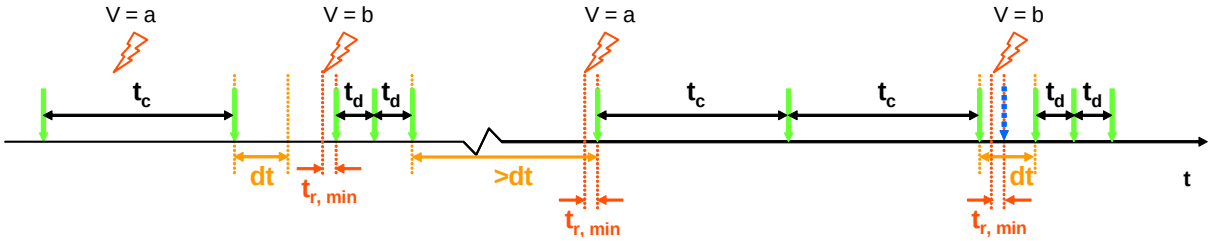

Figure 7: Use Case 4a, TM Periodic + Direct/N-Times, here n = 3 (with TMS switch)

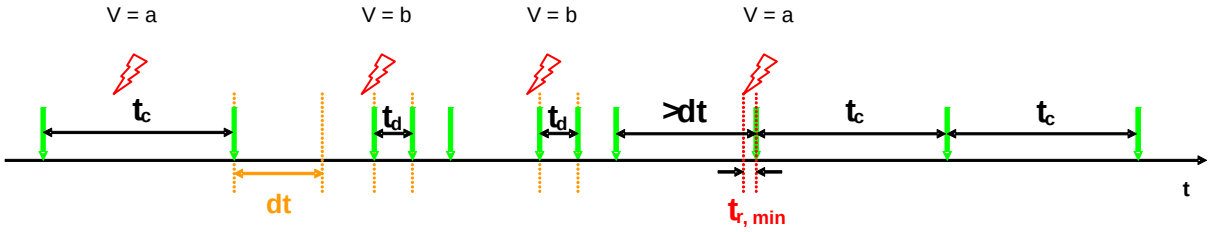

Figure 8: Use Case 4b, TM Periodic + Direct/N-Times, here n = 3 (with TMS switch)

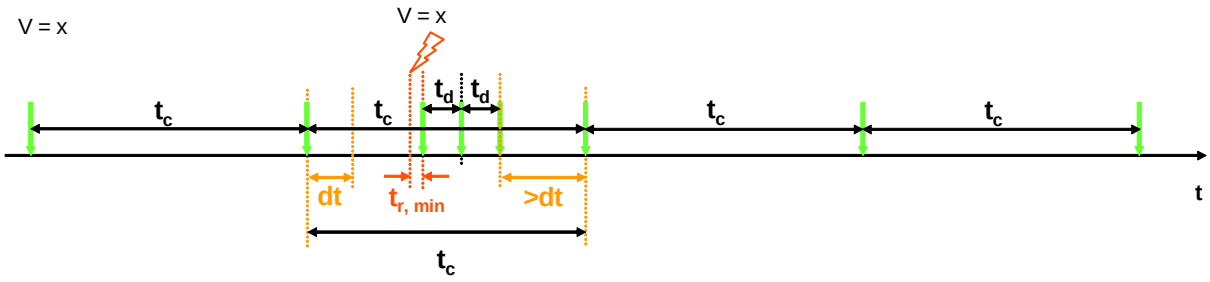

Figure 9: Use Case 5a, TM Mixed, here n = 3 (without TMS switch)

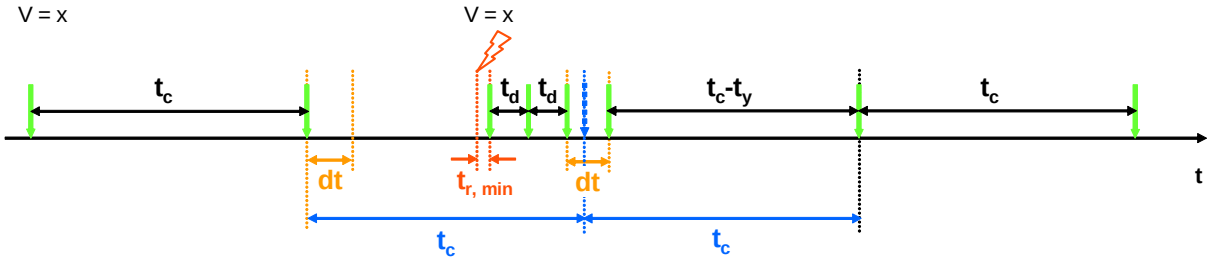

Figure 10: Use Case 5b, TM Mixed, here n = 3 (without TMS switch)

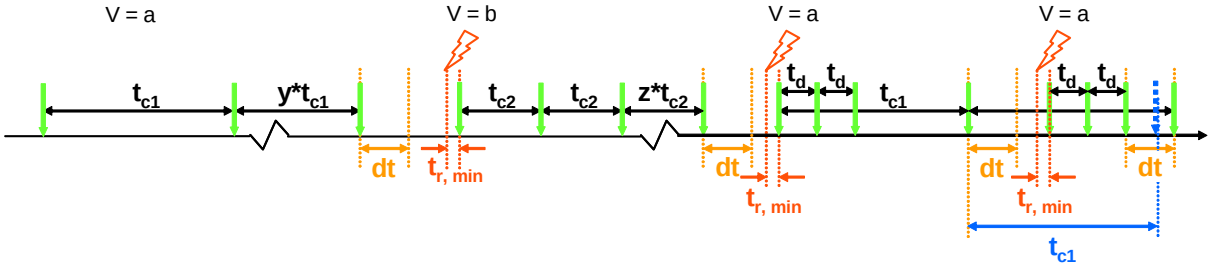

Figure 11: Use Case 6, TM Mixed, here n= 3 + Periodic (with TMS switch)

7.4.4 Signal invalidation mechanism

COM097: It shall be possible for the sender side to indicate AUTOSAR COM that it is not able to provide a valid value for a corresponding signal (e.g. sensor is faulty). It shall be possible during configuration to define a *Data Invalid Value* (for configuration see COM338).

7.4.4.1 Transmission of an invalidated signal

COM099: AUTOSAR COM shall replace the current value of a signal by the *Data Invalid Value* if the AUTOSAR Software Component indicates that it is not able to provide a valid value (`Com_InvalidateSignal()`). AUTOSAR COM shall internally perform a `Com_SendSignal()` with the configured *Data Invalid Value* (for configuration see COM447, COM316). Therefore the transmission of the *Data Invalid Value* on the bus is determined by the Transfer Property and the Transmission Mode.

Note: The internally performed Com_SendSignal with the *Data Invalid Value* lead to *Data Invalid Value* to be used as current value for filters and TMS.

COM286: By a call of `Com_InvalidateShadowSignal()` AUTOSAR COM shall replace the current value of the signal with the given SignalId within the associated signal group by the configured *Data Invalid Value*. By this call no send request shall be initiated.

Note: Compared to `Com_InvalidateSignal()`, there is no send request associated with `Com_InvalidateShadowSignal()`.

Note: The RTE has to call `Com_InvalidateShadowSignal()` for each signal inside a signal group.

Example with 2 signals signal_a and signal_b which belong to group_x:

```
/* invalidate signal a in shadow buffer with configured
data invalid value */
Com_InvalidateShadowSignal (signal_a);

/* invalidate signal b in shadow buffer with configured
data invalid value */
Com_InvalidateShadowSignal (signal_b);

/* copy shadow buffer to I-PDU */
Com_SendSignalGroup (group_x);
```

The *Data Invalid Values* are configured per signal (see COM447). For invalidating signal groups, there is no additional configuration necessary.

Note: The invalidation of the whole signal group results in a consistent state of the signal group during transmission. The following use cases require also a consistent state:

- If a gateway splits a signal group, the resulting signals / signal groups shall also be consistently invalidated.
- The VFB defines only one attribute for a complex data type. Therefore the best mapping of an invalidated complex data type to an invalidated signal group is to invalidate all signals of a signal group.

7.4.4.2 Reception of an invalidated signal

COM100: Receiver side AUTOSAR COM layer shall provide the following configuration options:

- **InvalidNotification**
  The reception of an invalidated signal shall be notified by the function described by COM315. The normal signal indication shall not take place.
- **ReplaceValue**
  When an invalidated signal is received, this signal value shall be replaced by the init value. Only the normal signal reception indication shall take place (if configured).

The configuration is described by COM338, COM314, COM315 and COM391.

COM287: The mechanisms as described in COM100 shall also be applicable for signal groups:

- **InvalidNotification**
  If at least one of the signals inside a signal group is identified as invalid, an invalid notification for the whole signal group shall take place. In this case, the normal signal group indication shall not take place.
- **ReplaceValue**
  If at least one of the signals inside a signal group is identified as invalid, all associated signals shall be replaced by the init value. In this case, only the normal signal group reception indication for this signal group shall take place.

COM323: For signals in signal groups it shall be possible to configure an invalid notification for the whole signal group and additionally for each single signal of the signal group.

Note: The use-case is if the receiver side is only interested in a single signal of signal group (receiver and sender side are independent). This is probably only a use case for the signal based gateway.

7.4.5 Handling of I-PDUs

7.4.5.1 Definitions

For the definition of an I-PDU group see [7] and Figure 12. For the configuration of I-PDU groups see COM206, COM126 and COM216.

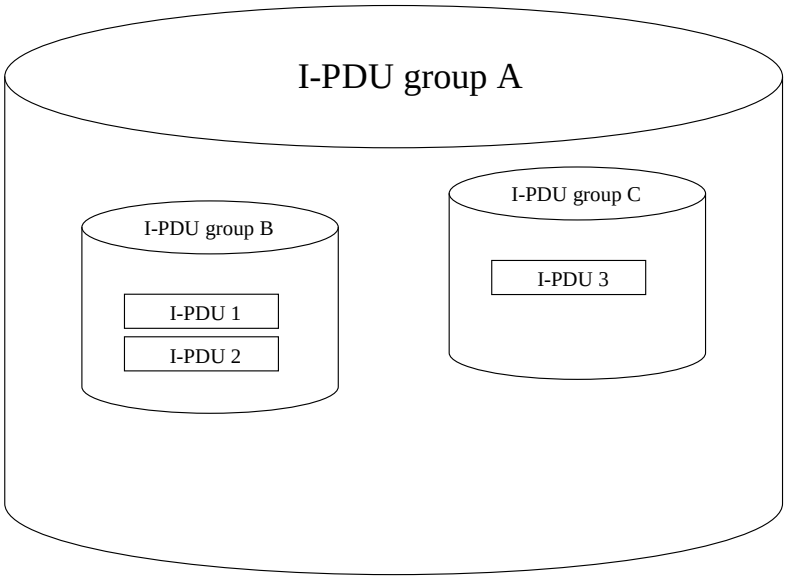

Figure 12: Grouping of I-PDUs and I-PDU groups

COM187: The number of supported I-PDU groups per ECU shall be limited to a maximum of 32 to simplify implementation.

7.4.5.2 Separate Start/Stop AUTOSAR COM for separate groups of I-PDUs

7.4.5.2.1 Starting of I-PDU groups

By default all I-PDU groups are stopped, see COM444.

COM085: AUTOSAR COM shall provide a routine which starts communication per I-PDU group. The name of the routine shall be `Com_IpduGroupStart()` (see COM191). A reference to the I-PDU group shall be transferred in the call to `Com_IpduGroupStart()`.

COM114: Starting an I-PDU group shall permit to transmit/receive I-PDUs which belong to the I-PDU group, see also Table 8.

COM222: If an I-PDU group is started by `Com_IpduGroupStart()` with parameter `Initialize` set to TRUE the following time attributes (see also Chapter 9.2) for each I-PDU belonging to the group shall be initialized:

- time period and offset attributes of I-PDUs in Periodic or Mixed Transmission Mode
- the minimum delay time attribute of I-PDUs in Direct/N-Times or Mixed Transmission Mode
- timeout attributes of I-PDUs for deadline monitoring aspect. All timeout timers (COM_NETWORK_SIGNAL_FIRST_TIMEOUT_FACTOR, COM_NETWORK_SIGNAL_TIMEOUT_FACTOR) shall restart

COM223: If an I-PDU group is started, the send mode of all associated I-PDUs is determined by the current data content of the I-PDU. E.g. transmission of periodic I-PDUs is started according to their TMS-switch.

Note: In order to disable transmission of a group of I-PDUs, the I-PDUs are put into an I-PDU group which is then stopped. This mechanism allows implementing listen-only-mode. Receiving of I-PDUs may be stopped also. Note that an I-PDU group cannot contain a mixture of Send I-PDUs and Receive I-PDUs.

COM228: When an I-PDU group containing an I-PDU is started but one or more signals in that I-PDU have not yet been written to via one of the Send-APIs, the initial value is used to determine the Send Mode.

COM229: When an I-PDU group containing an I-PDU is started but one or more signals in that I-PDU has been written to via one of the Send APIs by the higher layer, the most recently sent values are used to determine the *TMS of this I-PDU.*

COM476: If by starting an I-PDU group (or a contained I-PDU group) an already started I-PDU is started again the further starts shall be ignored.

7.4.5.2.2 Stopping of I-PDU groups

COM090: AUTOSAR COM shall provide a routine which supports the possibility of stopping communication per I-PDU group. The name of the routine shall be `Com_IpduGroupStop()`, see COM190. A reference to the I-PDU group shall be transferred in the call to `Com_IpduGroupStop()`.

COM334: If an I-PDU-group is stopped, a call of the functions: `Com_SendSignal()`, `Com_UpdateShadowSignal()`, `Com_SendSignalGroup()`, `Com_InvalidateSignal()` and `Com_InvalidateShadowSignal()` shall still update the values in the COM internal buffers, see Table 7.

Note: If a signal is written to an I-PDU that belongs to a stopped I-PDU group and this write would trigger the transmission of the I-PDU if it was not stopped then this trigger shall not be stored. After re-starting the corresponding I-PDU group such an old trigger shall not lead to an immediate transmission of the I-PDU.

COM115: Stopping I-PDU groups shall disable transmission of I-PDUs which belong to the I-PDU group. Only the `Com_TriggerTransmit()` function is processed because this can not be prohibited by AUTOSAR COM. Deadline monitoring for pending confirmations shall be cancelled. Transmit confirmations shall be ignored.
Stopping I-PDU groups shall disable processing of received I-PDUs which belong to the I-PDU group. Reception Deadline Monitoring shall be cancelled.

COM479: If an I-PDU group is stopped all outstanding not confirmed transmitted signals/ signal groups shall immediately receive an error notification if configured (see COM449).

COM461: A call to `Com_ReceiveSignalGroup()`shall always copy the last known data (or the init value) of the I PDU to the shadow buffer even if the I-PDU is stopped and COM_SERVICE_NOT_AVAILABLE is returned.

Note: If by stopping an I-PDU group (or a contained I-PDU group) an already stopped I-PDU is stopped again this has no effect.

Table 7 gives an overview of the behavior of stopped I-PDU groups:

Behavior on a stopped I PDU group	
Receiver side (RX)	***Transmitter side (TX)***
• Disable RX Deadline Monitoring • No action on a Com_RxIndication() to RTE, no storing of the I PDU • Return code COM_SERVICE_NOT_AVAILABLE on `Com_ReceiveSignal()`, `Com_ReceiveShadowSignal()` and `Com_ReceiveSignalGroup()` and the last known value (or init value) is given back as data	• Disable sending • Disable TX Deadline Monitoring • Com_TxConfirmation(): if it is for timeout ignore it if it is used by the RTE ignore it. • On a call of `Com_SendSignal()`, `Com_UpdateShadowSignal()`, `Com_SendSignalGroup()`, `Com_InvalidateSignal()` and `Com_InvalidateShadowSignal()`

	the values in the COM internal buffers are still up-dated • Outstanding transmission request (e.g. N-Times) shall be cancelled • Normal reaction on `Com_TriggerTransmit()`
	For periodic (TX)
	Do not send any more

Table 7: Behavior of stopped I-PDU-groups

Table 8 gives an overview of the behavior of started I-PDU groups:

Behavior on a started I PDU group	
Receiver side (RX)	***Transmitter side (TX)***
• Reinitialize timeouts if `Initialize==` TRUE (COM_NETWORK_SIGNAL_FIRST_TIMEOUT_FACTOR, COM_NETWORK_SIGNAL _TIMEOUT) • Normal reaction on `Com_RxIndication()` • Normal reaction on `Com_ReceiveSignal()`, `Com_ReceiveShadowSignal()` and `Com_ReceiveSignalGroup()`	• Normal reaction on `Com_InvalidateSignal()`, `Com_InvalidateShadowSignal()`, `Com_SendSignal()`, `Com_SendSignalGroup()`, `Com_UpdateShadowSignal()` and `Com_SendSignalGroup()` • No transmission timeout notification until next send • Normal reaction on `Com_TxConfirmation()` • Normal reaction on `Com_TriggerTransmit()`
	For periodic (TX)
	Start at 0

Table 8 Behavior of started IPDU-groups

COM311: When an I-PDU group containing one or more other I-PDU groups is started the contained I-PDU groups shall also be started. When an I-PDU group containing one or more other I-PDU groups is stopped the contained I-PDU groups shall also be stopped.

Note: According to [7] there is only a two level hierarchy of I-PDU groups. That is an I-PDU-group contained in another I-PDU group is not allowed to have any further subgroups. As example see also Figure 12.

With respect to Figure 12 the following examples are given:

- If A is stopped then all I-PDUs are stopped
- If A is started then all I-PDUs are started.
- If A is stopped and then B is started only I-PDU 1 and I-PDU 2 are active.
- If A is started and then B is stopped only I-PDU 3 is active.

7.4.5.3 Signal indication (Unpacking of I-PDUs)

COM298: In order to support both interrupt-driven and polled system, it shall be configurable when the signal indication takes place. There shall be two configurable signal indication modes:

IMMEDIATE:
The signal indications / confirmations are performed in `Com_RxIndication()`/ `Com_TxConfirmation()`

DEFERRED:
Signal indication / confirmations are deferred for example to a cyclic task

COM300: In IMMEDIATE mode the signal notification shall be done in `Com_RxIndication()`.

COM301: In DEFERRED mode: First in `Com_RxIndication()` the I-PDU's data shall be copied from the underlying layer into COM. Then `Com_RxIndication()` shall return. Then the signal notification shall be processed asynchronously, for example during the next call to `Com_MainFunctionRx()`.

Note: If in DEFERRED mode a call to `Com_ReceiveSignal()` is made before the deferred unpacking takes place the previous (not updated) values are returned.

A sequence chart with both indication options can be found in Chapter 9.3. The configuration of these modes is defined in COM119.

7.4.5.4 Minimum Delay Timer (MDT)

The minimum delay timer shall be defined as in [17].

Note: When an I-PDU group is started the MDT is re-initialized (see COM222). Therefore the MDT can be violated by stopping and starting I-PDU groups rapidly.

Note: As defined in [17] the MDT is started when the confirmation of the sent I-PDU is received. Therefore exists a time-slot between the send-call and the confirmation where an (erroneous) application could create a burst of I-PDUs by sending rapidly.

Note: The behavior of the Transmission Deadline Monitoring timer shall not be affected by any transmission delay caused by the minimum delay time supervision.

7.4.6 Deadline Monitoring

In the context of deadline monitoring a signal group is always handled like a signal. The deadline monitoring parameters can be configured per COM_NETWORK_-SIGNAL object (see Chapter 10.2.2.8), thus it is valid for single signals and for signal groups (treated as a signal).

For a configuration of COM_NETWORK_SIGNAL_FIRST_TIMEOUT_FACTOR see COM183, for the COM_NETWORK_SIGNAL_TIMEOUT_FACTOR see COM263, and for COM_NOTIFICATION see COM343.

7.4.6.1 Reception Deadline Monitoring

COM292: In the case where deadline monitoring is used on signals with Update-bits, there shall be a separate Reception Deadline Monitoring for each signal with an update bit. For configuration see COM263 and COM264.

COM290: There shall be an I-PDU based Reception Deadline Monitoring for signals without an update bit.

COM291: For all signals without Update-bits within the same I-PDU, the smallest configured timeout parameter (COM_NETWORK_SIGNAL_FIRST_TIMEOUT_-FACTOR and COM_NETWORK_SIGNAL_TIMEOUT_FACTOR) of the associated signals is chosen as timeout parameter for the Reception Deadline Monitoring of the I-PDU.

Note: If all signals have an update bit in an I-PDU, no Reception Deadline Monitoring on I-PDU base needs to be performed.

COM393: In case of an Rx-timeout it shall be configurable whether COM replaces the signal / signal group value with the initial value or keeps it as it is. In case of replacement the *old_value* of the corresponding filter-object (if configured) shall not be replaced.

Note: Rx-timeout-indication can be combined and configured separately from COM393.

7.4.6.1.1 En-/Disable Reception Deadline Monitoring

COM092: AUTOSAR COM shall provide a routine which supports the possibility to disable Reception Deadline Monitoring for an I-PDU group. The name of the routine shall be `Com_DisableReceptionDM()` (See COM192). A reference to the I-PDU group shall be transferred in the call to `Com_DisableReceptionDM()`.

Disabling Reception Deadline Monitoring implies that no error indication of deadline monitoring expiry is notified to the upper layer for an I-PDU in reception belonging to the concerned I-PDU group.

Disabling Reception Deadline Monitoring shall not stop communication activity for the concerned I-PDU group.

COM095: AUTOSAR COM shall provide a routine which supports the possibility to re-enable Reception Deadline Monitoring for an I-PDU group where Reception Deadline monitoring has been previously disabled. The name of the routine shall be `Com_EnableReceptionDM()` (See COM193). A reference to the I-PDU group shall be transferred in the call to `Com_EnableReceptionDM()`.

COM224: A call to `Com_EnableReceptionDM()` shall reset the Reception Deadline Monitoring timer to COM_NETWORK_SIGNAL_FIRST_TIMEOUT_FACTOR if and only if Reception Deadline Monitoring was disabled for the corresponding I-PDU group.

COM486: Enabling an already enabled deadline monitoring shall have no effect.

Enabling Reception Deadline Monitoring implies that error indication of deadline monitoring expiry is notified to the upper layer for an I-PDU in reception belonging to the concerned I-PDU group.

COM225: Disabling an already disabled deadline monitoring shall have no effect.

7.4.6.2 Transmission Deadline Monitoring

For Transmission Deadline Monitoring there is no difference between signals with Update-bits and signals without Update-bits.
Therefore Transmission Deadline Monitoring can be performed on I-PDU base.
Nevertheless notification about detected deadline violations on sender side is done per signal. See [17] for further details.

COM481: Transmission Deadline Monitoring shall not distinguish between signals with Pending or Triggered Transfer Property. It shall be performed for all signals and all Transmission Modes as for signals with Triggered Transfer Property in [17] if configured.

COM445: If different COM_NETWORK_SIGNAL_TIMEOUT_FACTOR parameters (see COM263) of the associated signals of an I-PDU are configured the smallest values shall be used as COM_NETWORK_SIGNAL_TIMEOUT_FACTOR parameter for the Transmission Deadline Monitoring of the I-PDU.

Note: Transmission Deadline Monitoring should only be configured in COM if the lower layer supports the generation of transmit confirmations. Otherwise the Transmission Deadline Monitoring would always notify a transmission error.

Note: The PDUR does not support transmit confirmations for PDUs that are configured to be fanned out by the PDUR to multiple receivers.

COM458: In case of a signal group (see section 7.5) it shall only be possible to configure Transmission Deadline Monitoring for the whole signal group and not for individual signals of the signal group.

7.4.6.2.1 Clarification of the OSEK COM specification

The following requirement COM304 states the behavior of the Transmission Deadline Monitoring in the Mixed Transmission Mode defined in [17] more precisely.

COM304: If the transmission does not occur, i.e. if there is no confirmation of the I-PDU's transmission by the underlying layer, the time-out occurs and the application shall be notified using the appropriate notification mechanism.

Note: In the case that there are any contradictions between text and diagrams in [17] the text is the normative part.

7.4.6.2.2 Transmission Deadline Monitoring with N-Times Transmission Mode

As defined in [17] the monitoring timer is started upon completion of a call to the `Com_SendSignal()` API service.

COM307: In Direct/N-Times Transmission Mode it must be ensured that all n requests can be made within the configured time period, see Chapter 7.4.3.5.

As defined in [17], if the monitoring timer expires the upper layer is notified with the configured notification mechanism about that failure.

COM308: In Direct/N-Times Transmission Mode with n > 0 the timer shall only be cancelled when n-confirmations are received or if another send request for this I-PDU is initiated.

Note: If the timer is cancelled after the n-th confirmation the transmission was successful and then the transmission confirmation is send to the upper layer. See also COM305.

7.5 Map Complex Data Types to I-PDUs – Signal Groups

COM042: To support the AUTOSAR concept of complex data types the AUTOSAR COM layer provides signal groups (see Chapter 2). A signal group shall be transmitted and received atomically; therefore it provides data consistency for complex data types.

COM043: Signal groups are configured statically. For each group a symbolic name shall be provided. See COM044 and COM045 for the configuration details.

COM047: The atomicity of a signal group shall be achieved by means of a shadow buffer mechanism, i.e. the upper layer uses the signals in the shadow buffer. If the shadow buffer needs to be synchronized with the I-PDU this is triggered explicitly by the upper layer. Synchronization shall be performed atomically.

7.5.1 Initialization

COM484: The shadow buffer of a signal group on sender-side shall be initialized by a call to `Com_Init()`.

Note: Since it is not suspected that a well-formed SWC tries to read the signal of a signal group in the receiver call before a call to `Com_ReceiveSignalGroup()` COM484 applies only to the sender side.

7.5.2 Transmission

COM049: A group signal in the shadow buffer shall be updated by the call of the service `Com_UpdateShadowSignal()`.

COM050: If `Com_SendSignalGroup()` is called for the signal group the AUTOSAR COM layer shall copy the shadow buffer atomically to the I-PDU buffer.

Example with 2 signals signal_a and signal_b which belong to group_x:

```
/* copy a to shadow buffer */
Com_UpdateShadowSignal (signal_a, &a);

/* copy b to shadow buffer */
Com_UpdateShadowSignal (signal_b, &b);

/* copy shadow buffer to I-PDU */
Com_SendSignalGroup (group_x);
```

COM462: For individual signals of a signal group the configuration item COM_SIGNAL_TRANSFER_PROPERTY (see COM323) shall be ignored. The Transfer Property of the signal group shall be configured via COM_SIGNAL_GROUP_TRANSFER_PROPERTY (see COM350).

7.5.3 Reception

COM051: If `Com_ReceiveSignalGroup()` is called for the signal group the AUTOSAR COM layer shall copy the data from the I-PDU buffer to the shadow buffer.

COM052: A signal of a signal group shall be received from the shadow buffer by means of the service `Com_ReceiveShadowSignal()`.

Example with 2 signals signal_a and signal_b which belong to group_x:

```
/* copy I-PDU to shadow buffer */
Com_ReceiveSignalGroup (group_x);

/* copy a from shadow buffer */
Com_ReceiveShadowSignal (signal_a, &a);

/* copy b from shadow buffer */
Com_ReceiveShadowSignal (signal_b, &b);
```

7.5.4 Notifications

COM053: It shall only be possible to configure the signal and error notifications (on sender and receiver side) for the whole signal group. The notifications shall be sent to the RTE if a whole signal group has been sent to / received or detected to be invalid from the lower layer. See Chapter 10.2.2.13 for the configuration details.

7.5.5 Collection of the attributes of a signal group

Table 9 gives an overview of the attributes of a signal group:

Attribute	***Per signal inside a signal group***	***Per signal group***
Update bit	No	Yes, associated on the whole group (see 7.7)
Signal Notification (sender side)	No	Yes
Signal Notification (receiver side)	No	Yes
Error Notification (sender side)	No	Yes
Error Notification (receiver side)	No	Yes
Invalid Notification (receiver side)	Yes (see COM323)	Yes (see COM323)
Data access (receiver side)	Yes, with `ReceiveSignal()` API (see 8.3.2.2)	Yes, via shadow buffer (see 8.3.2.5 and 8.3.2.6)
Data access (sender side)	No	Yes, via shadow buffer (see 8.3.2.3 and 8.3.2.4)
Data Filtering (receiver side)	No (see 7.2.3)	No
Data Filtering (sender side)	No	No
TMS on sender side	Each signal, according to TMS selection definition. (see 7.4.3.3)	No

Table 9: Attributes of signal groups

7.6 Interface between AUTOSAR COM and the lower layer (PDU-Router)

OSEK COM leaves the interface between OSEK COM and the lower layers undefined. In AUTOSAR the only lower layer that AUTOSAR COM interfaces to is the PDU Router.

The interaction diagram in Chapter 9.1 shows the interaction between the PDU Router and AUTOSAR COM.

A detailed description can be found in the API chapter (see `Com_RxIndication()`, `Com_TxConfirmation()` and `Com_TriggerTransmit()`).

COM138: When AUTOSAR COM wants to send out an I-PDU, AUTOSAR COM shall use the `PduR_ComTransmit()` function.

7.7 Signal status information

7.7.1 Identify if a signal is updated by the sender

COM054: To enable the receiver of a signal/ signal group to identify whether the sender has updated the data in this signal/ signal group before sending, AUTOSAR COM shall support *Update-bits*.

The Update-Bit shall indicate whether upper layers on sender-side have updated a signal value before lower layers have fetched the I-PDU containing that signal for transmission or before AUTOSAR COM has handed over the I-PDU to lower layers for transmission.

Note: Update-bits are not allowed if Direct/N-Times Transmission Mode with n>1 is used (see COM310).

COM116: Update-bits shall only be supported for external communication.

COM055: By configuration on sender- and on receiver-side, it shall be possible to add separately for each signal and/or separately for each signal group at most one additional bit (= update-bit). The update-bit is not part of the signal or signal group itself and shall only be used by AUTOSAR COM itself. The update bit shall not be visible to or accessible by the AUTOSAR Software Component.

COM056:´The position of the update bit shall be configurable.
For configuration parameter see COM257.

COM057: Signal / signal group and the corresponding update bit shall always be part of the same I-PDU.

COM059: The interpretation of the update bit shall be as follows:

Update BIT	
0	cleared
1	set

Table 10 update bit interpretation

If the value of the update-bit is set, data has been updated; if the value of the update-bit is cleared it has not been updated.

7.7.1.1 Sender Side

The initialization of the update bit is defined in the chapter *Start-up* by COM117.

COM061: If upper layers update the value of a signal by calling the AUTOSAR COM API `Com_SendSignal()`, the update-bit for this signal shall be set. For signal groups, the update bit shall be set, if the upper layers call the AUTOSAR COM API `Com_SendSignalGroup()`.

COM062: After an I-PDU is sent to lower layers and no synchronous error is returned by the lower layer the Update-bits of all signals and signal groups belonging to the I-PDU sent shall be cleared (the update-bit will be set again if the AUTOSAR Software Component updates the signal, see COM061).

7.7.1.2 Receiver Side

COM324: On receiver-side, if there is an update-bit attached to a signal/signal group, AUTOSAR COM shall only process this signal (i.e. filter, notification, signal based, byte swapping), if the signal has been updated. If the signal has not been updated AUTOSAR COM shall discard the signal.

Note: If the signals has not been updated the signal will not be routed via the signal gateway. It will only be discarded.

Remark: If the upper layer reads a signal with an associated cleared update bit, the init value or the last received value is returned.

COM067: A signal/signal group shall be interpreted as *updated* if the signal has an update-bit attached and the value of the update bit is set.

COM076: After the `Com_IpduGroupStart()` API of AUTOSAR COM has been processed all related Update-bits are cleared with `Com_IpduGroupStart()`.

For the behavior of deadline monitoring on signals with Update-bits, see Chapter 7.4.5.4.

7.8 Callouts

As stated in COM013 *Network-order Message Callout* and *CPU-order Message Callout* are not supported in AUTOSAR COM. The only callout method in AUTOSAR COM therefore is the I-PDU-Callout. AUTOSAR COM supports I-PDU-Callouts on sender and on receiver side.

7.8.1 I-PDU Callout

COM381: In an IPDU-callout other COM APIs than `Com_TriggerIPDUSend()` shall not be called.

COM346: The IPDU-Callout API shall be defined as:

```
boolean <IPDU_CalloutName>(PduIdType ID, uint8* ipduD)
```

where <IPDU_CalloutName> has to be substituted with the concrete IPDU-callout name.

Note: As specified in OSEK COM if the I-PDU-Callout returns false the I-PDU shall not be processed any further.

COM347: It shall be possible to configure separate IPDU-callout function for each I-PDU. Therefore the IPDU-callout function shall be configurable per I-PDU if used.

For configuration see COM387.

COM395: When `Com_TriggerTransmit()` is called, COM shall ignore the return value from the I-PDU callout (if configured).

7.9 Signal Gateway

The signal gateway is an integrated part of COM. The signal gateway can't be accessed by any external modules, except the cyclic task call.

The signal gateway is working on network signals (containing signals or signal groups).

COM376: The Signal Gateway only supports static routing: All routes shall be used independent of the content of the signals and signal groups to be routed. The destination of a signal or signal group shall only depend on the name of the received signal respectively signal group.

COM377: The Signal Gateway shall copy the value of signals respectively signal groups to be routed to the signals respectively signal groups for transmission according to configuration.

COM358: It shall be possible to route a signal/ signal group from one source signal/ signal group to zero (no signal gateway functionality) or more destinations (1:n).

Note: COM358 applies also for signal/ signal groups that are received locally. That means a signal/ signal group can be locally received and routed.

7.9.1 Dealing with signals

COM357: COM shall forward signals to be routed from received I-PDUs to transmit I-PDUs. For configuration see COM356 and COM382.

COM360: If the endianness of a received signal to be routed differs from the endianness of a related destination signal, its endianness shall be converted using the applicable COM mechanisms.

7.9.2 Dealing with signal groups

COM361: COM shall forward signals groups to be routed from received I-PDUs to transmit I-PDUs. For configuration see COM356.

COM383: Signal groups shall be routed in an atomic manner (i.e. it must be guaranteed that the data within the signal group is transferred as one consistent set of data during the whole routing operation).

COM362: It shall be possible to change the endianness of signals contained in signal groups to be routed by Signal Gateway. The Signal Gateway shall use the applicable COM mechanisms to do so.

COM464: All non-opaque signals within a signal group shall have the same endianness.

7.9.3 Routing of out timed signals and signal groups

In case of a not in time received signal or signal group (timeout of the related supervising timer), this signal or signal group reception timeout shall be ignored by the signal gateway.

7.9.4 Decoupling signal gateway

To protect interrupt routines (used for I-PDU reception) from incalculable (and perhaps expensive) time usage, it is necessary to decouple the signal gateway from interrupt routines.

COM359: The functions of the signal gateway shall be executed during a separate function call only. During this function call the Signal Gateway checks received and to be routed signals and signal groups and forwards them from the related receive I-PDUs to the related transmit I-PDUs (see COM400).

COM466: Within `Com_MainFunctionRouteSignals()` the evaluation of Transfer Properties and Transmission Mode must be performed in the following sequence (see also Figure 3):

1. Copy all gatewayed signals from the source to the target I-PDUs
2. Evaluate the TMC of all gatewayed signals
3. Evaluate the TMS for the target I-PDUs
4. For any target I-PDU containing gatewayed signals with Triggered Transfer Property send it according to its Transmission Mode

7.10 Error classification

7.10.1 Development Errors

COM024: All input parameters shall be checked for validity during development. The parameter check shall be not contained in the production code. For the configuration of this feature see COM028.

COM442: When a development error is detected the function `Det_ReportError()` of the development error tracer shall be called with:

- the COM moduleID (see COM417)
- the service ID of the COM API the error is detected (see `Com_ServiceIdType`)
- the error ID as defined in Table 11

Type of development error	*Related error code*	*Value [hex]*
API service called with wrong parameter	COM_E_PARAM	0x01
Error code if any other API service is called before COM was initialized with `Com_Init()` or after a call to `Com_Deinit()`	COM_E_UNINIT	0x02

Table 11: Mapping of COM development error IDs

7.10.2 Production Errors

Actually no production errors are defined in AUTOSAR COM. If production errors will be defined in later versions AUTOSAR COM shall report them directly to the DEM.

7.10.3 Return Codes

AUTOSAR COM does not define a special COM return type. The API services return errors either by using the Std_ReturnType as defined in [5] or via a uint8 value mapped according to Table 12.

COM459: Return codes of AUTOSAR Com shall be defined according Table 12.

Name	*Description*	*Type*	*Value*	*Defined in*
E_OK	the service has been accepted	`#define`	`0x00`	`Std_Types.h`
E_NOT_OK	a development error has been detected	`#define`	`0x01`	`Std_Types.h`
COM_SERVICE_NOT_AVAILABLE	the service is currently not available e.g. the corresponding I-PDU group is stopped	`#define`	`0x80`	`Com.h`
COM_TIMEOUT	a timeout has occurred	`#define`	`0x81`	`Com.h`

Table 12: Mapping of AUTOSAR COM return codes

7.11 Error handling

COM428: If not stated otherwise AUTOSAR COM will ignore all errors from the underlying communication layer.

Note: AUTOSAR COM supervises the communication with deadline monitoring if configured. The specific error codes from the underlying layer therefore can be ignored. In case of Update-bits this error codes are handled see COM062.

7.12 AUTOSAR COM interaction model

This chapter corresponds to the chapter *Functional Model of Interaction Layer* of [17]. The following figures illustrate the behavior of the Interaction Layer for external reception and external transmission. The complete functionality is shown but it depends on the configuration what parts are present/ used in a concrete implementation.

COM396: A signal can be configured to have filtering, data invalidation and notification. The order of execution (if configured) is:

1) Filtering
2) Data Invalidation
3) Notification

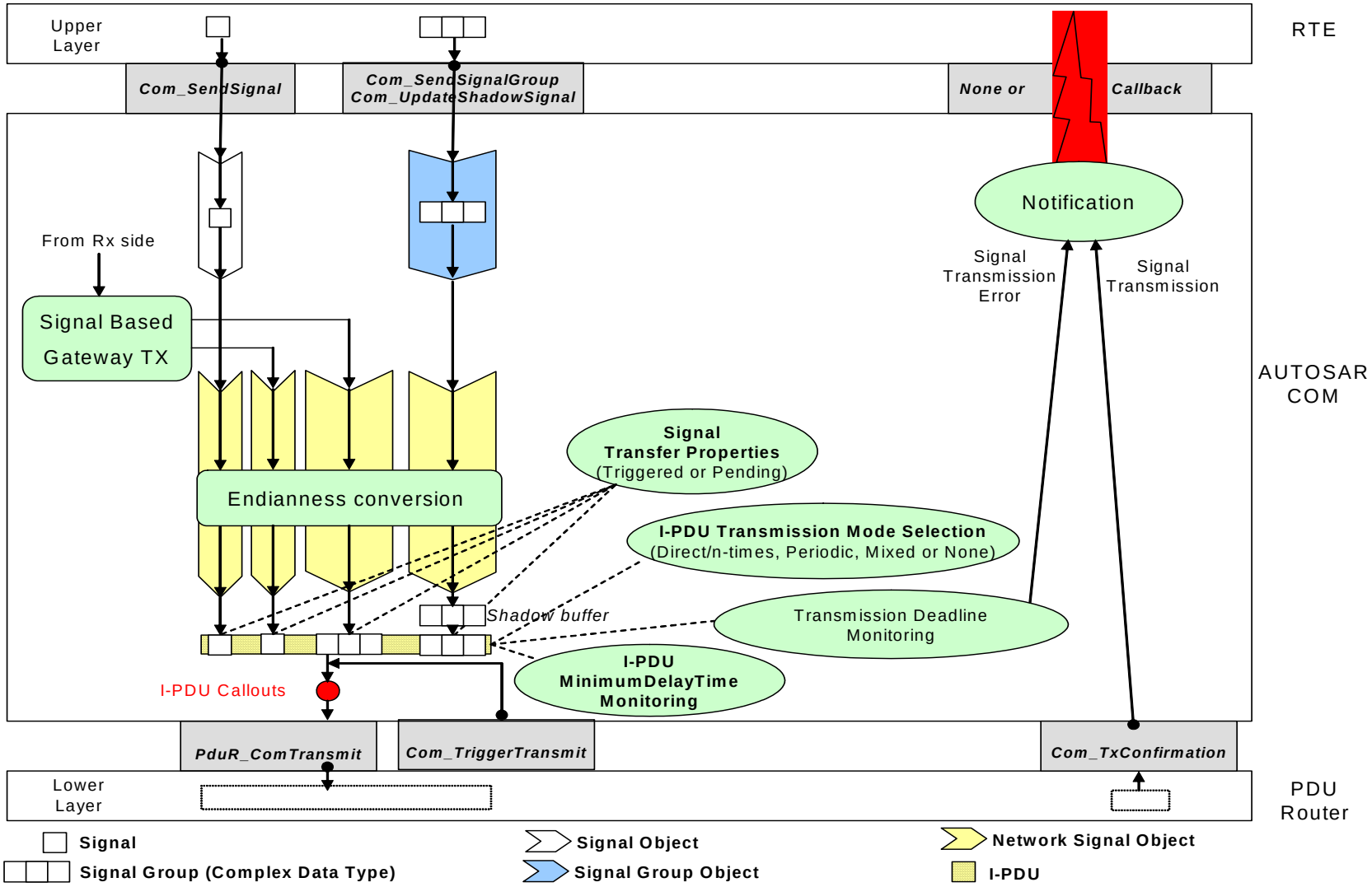

Figure 13 AUTOSAR COM interaction model for external transmission

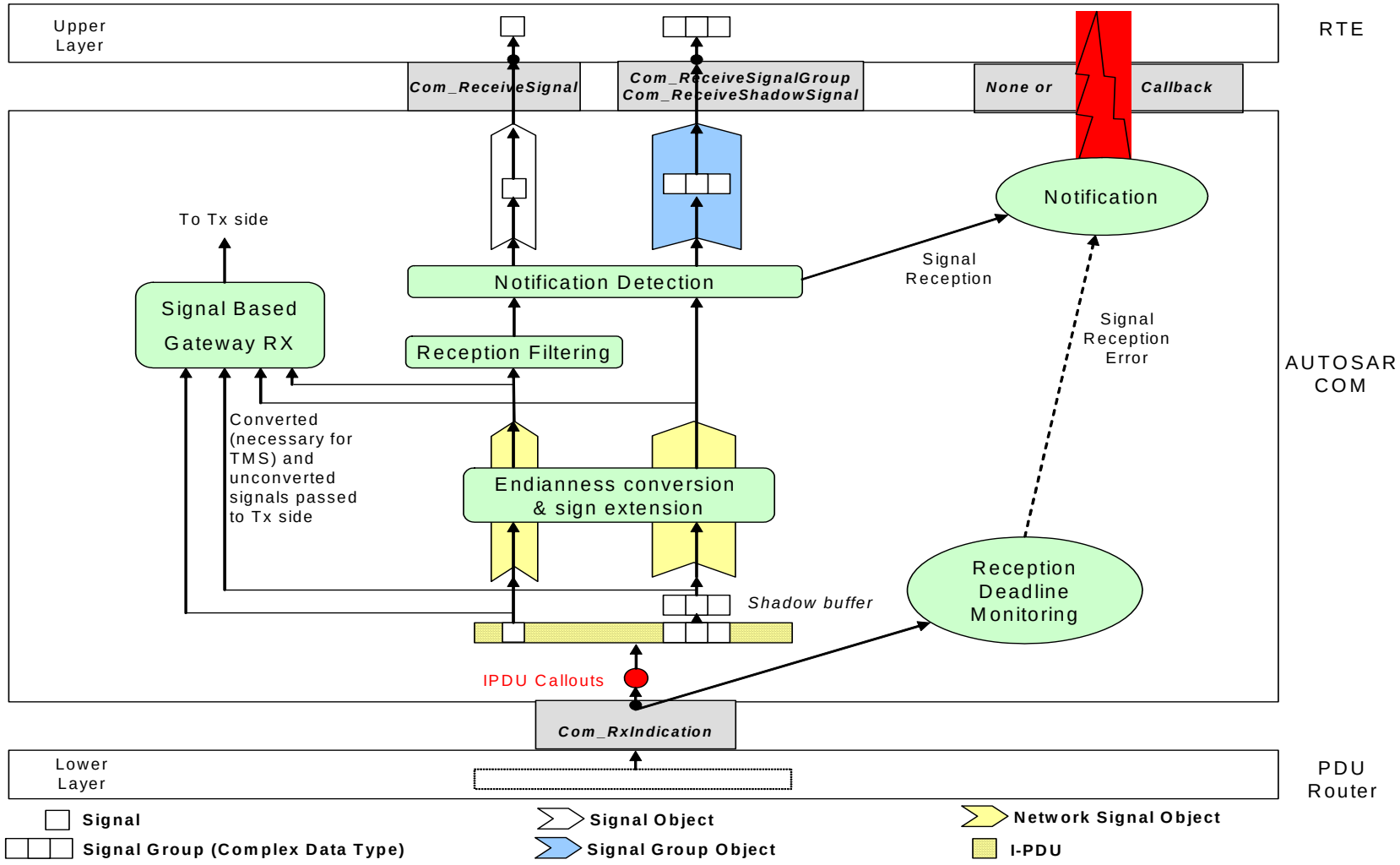

Figure 14 AUTOSAR COM interaction model for external reception

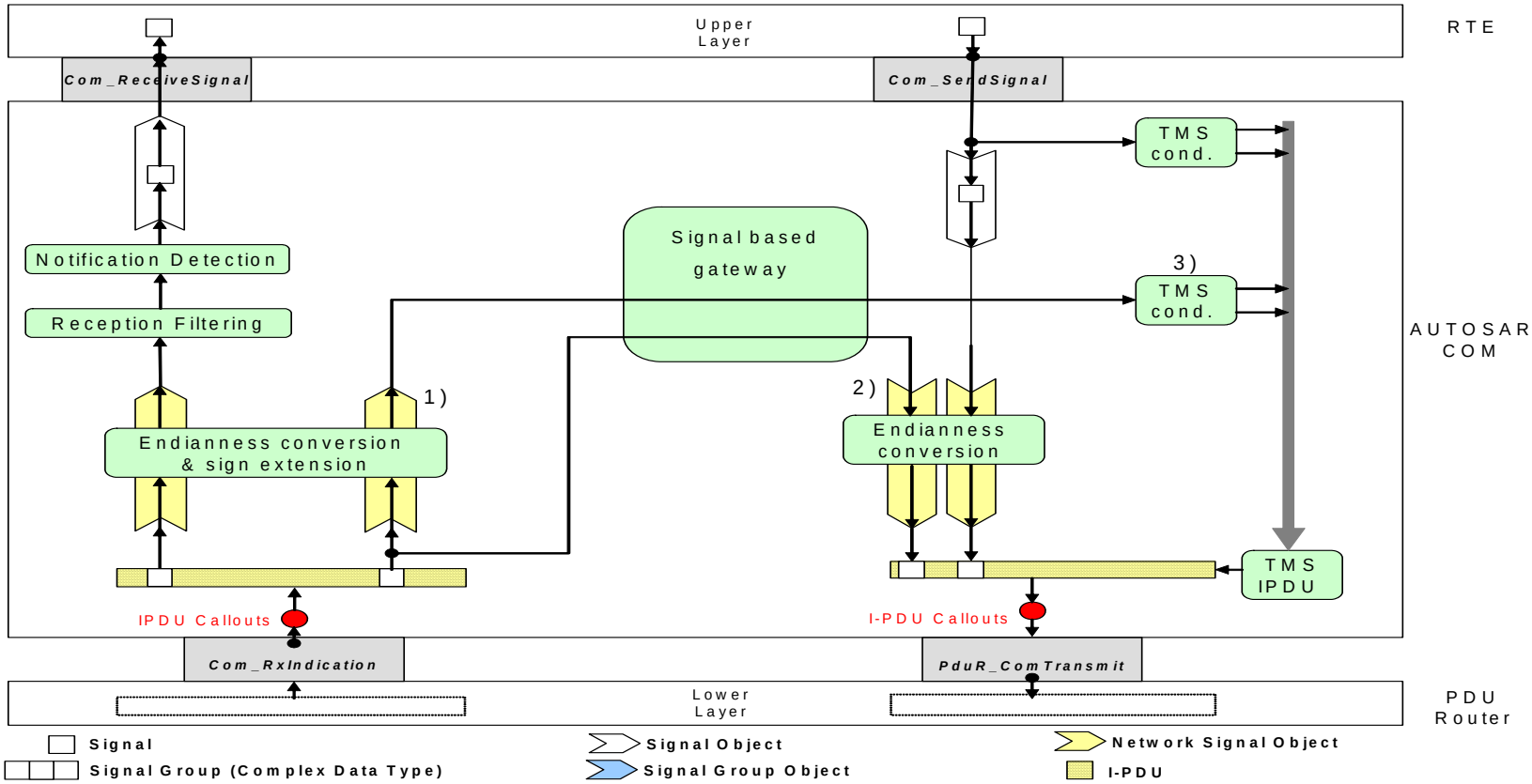

Figure 15: AUTOSAR COM-interaction model for Signal Gateway

The endianness conversion and sign extension on receiver side is needed to feed the TMS with a correct data format. This endianness conversion is only necessary if the endianness of the Rx-bus differs from the endianness of the CPU. The endianness conver-

sion on the sender side is only necessary if the endianness of the Rx-bus differs from the endianness of the Tx-bus. If a gated signal should not always trigger the associated IPDU to be sent out, one of the configured Transmission Modes must be none.

8 API specification

8.1 Imported types

8.1.1 PduIdType

The definition of the PduIdType can be found in [2].

8.1.2 Std_ReturnType

The definition of the Std_ReturnType can be found in [5].

8.1.3 Std_VersionInfoType

The definition of the Std_VersionInfoType can be found in [5].

8.2 Type definitions

8.2.1 Com_StatusType

Type:	Enum	
Range:	`COM_UNINIT`	The AUTOSAR COM module is not initialized or not usable. This shall be the default value after reset. This status shall have the value 0.
	`COM_INIT`	The AUTOSAR COM Module is initialized and usable.
Description:	This is a status value returned by the API service `Com_GetStatus()`.	

8.2.2 Com_SignalIdType

Type:	uint16	
Range:	`0 … <SignalId-max>`	Zero-based integer number
Description:	AUTOSAR COM signal object identifier.	

8.2.3 Com_SignalGroupIdType

Type:	uint8	
Range:	`0 … <SignalGroupId-max>`	Zero-based integer number
Description:	AUTOSAR COM signal group object identifier.	

8.2.4 Com_ApplicationDataRefType

Type:	void *
Range:	`Not applicable`
Description:	Pointer to the data field of the RTE.

8.2.5 Com_PduGroupIdType

Type:	uint8
Range:	`0 … <PduGrou-plIdmax>` Zero-based integer number
Description:	AUTOSAR COM PDU group object identifier.

8.2.6 Com_ServiceIdType

Type:	enum		
Range:	COMServiceId_xx with xx being the name of an AUTOSAR COM service as stated right. No additional leading Com_ is included, see example below.	Init	0x01
		DeInit	0x02
		IpduGroupStart	0x03
		IpduGroupStop	0x04
		DisableReceptionDM	0x05
		EnableReceptionDM	0x06
		GetStatus	0x07
		GetConfigurationId	0x08
		GetVersionInfo	0x09
		SendSignal	0x0A
		ReceiveSignal	0x0B
		UpdateShadowSignal	0x0C
		SendSignalGroup	0x0D
		ReceiveSignalGroup	0x0E
		ReceiveShadowSignal	0x0F
		InvalidateSignal	0x10
		ErrorGetServiceId	0x11
		Error_<Name1>_<Name2> Macros	0x12
		TriggerTransmit	0x13
		RxIndication	0x14
		TxConfirmation	0x15
		InvalidateShadowSignal	0x16
		TriggerIPDUSend	0x17
		MainFunctionRx	0x18
		MainFunctionTx	0x19
		MainFunctionRouteSignals	0x1A
Description:	Unique identifier of an AUTOSAR COM service. Example: *COMServiceId_SendSignal 0x0A*.		

8.2.7 Com_ConfigType

Type:	structure	
Range:	implementation specific	The content of the initialization data structure is implementation specific
Description:	This is the type of the data structure containing the initialization data for COM.	

8.3 Function definitions

COM320: If a function is marked as non-reentrant the caller of that function shall ensure that this function must not be called while it is running.

COM321: Non-reentrant functions do not have to check if they are called reentrant.

COM434: It is allowed to use macros instead of functions where source code is used and runtime is critical.

8.3.1 Start up and control services

8.3.1.1 Com_Init

COM432:

Service name:	Com_Init	
Syntax:	`void Com_Init ( const Com_ConfigType* config )`	
Service ID [hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Non reentrant	
Parameters (in):	config	Pointer to the COM configuration data.
Parameters (out):	None	
Return value:	None	
Description:	This service initializes internal and external interfaces and variables of the AUTOSAR COM Layer for the further processing. After calling this function the ECU internal communication is activated and possible, but the inter-ECU communication is still disabled. For further details see Chapter 7.4.1. COM433: If the config parameter does not correspond to a valid configuration then the development error `COM_E_PARAM` is generated. The behavior of AUTOSAR COM is unspecified until a correct call to `Com_Init()` is made.	
Caveats:	`COM_Init()` shall not pre-empt any COM function. The rest of the system must guarantee that COM_Init is not called in such a way.	
Configuration:	--	

8.3.1.2 Com_DeInit

COM130:

Service name:	Com_DeInit	
Syntax:	`void Com_DeInit ( void )`	
Service ID [hex]:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Non reentrant	
Parameters (in):	None	
Parameters (out):	None	
Return value:	None	
Description:	This service stops the ECU internal and the inter-ECU communication. All started I-PDU groups (see `Com_IpduGroupStart()` /`Com_IpduGroupStop()`) are stopped and have to be started again, if needed, after `Com_Init()` is called. By a call to ComDeInit COM is put into an not initialized state. For further details see Chapter 7.4.1.	
Caveats:	`COM_DeInit()` `shall` not pre-empt any COM function. The rest of the system must guarantee that COM_DeInit is not called in such a way.	
Configuration:	--	

8.3.1.3 Com_IpduGroupStart

COM191:

Service name:	Com_IpduGroupStart	
Syntax:	`void Com_IpduGroupStart ( Com_PduGroupIdType IpduGroupId, ...boolean Initialize )`	
Service ID [hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different I-PDU groups. Non reentrant for the same I-PDU group.	
Parameters (in):	`IpduGroupId`	Id of I-PDU group to be started
	`Initialize`	flag to request initialization of the data in the I-PDUs of this I-PDU group
Parameters (out):	None	
Return value:	None	
Description:	Starts a preconfigured I-PDU group. For example, cyclic I-PDUs will be sent out cyclically after the call of `Com_IpduGroupStart()`. See Chapter 7.4.5 for details. If `Initialize` is `TRUE` all I-PDUs of the I-PDU group shall be (re-)initialized before the I-PDU group is started. That is they shall behave like after a start-up of COM, for example the old_value of the filter objects and shadow buffers of signal groups have to be (re-)initialized.	
Caveats:	A call to `COM_IpduGroupStart()` shall not be interrupted by another call to	

	`COM_IpduGroupStart()` or a call to `COM_IpduGroupStop()`. Note that this function is not only called by the COMM but also from other modules e.g. diagnosis.
Configuration:	An I-PDU group must be configured before this call. See COM206 and COM341 for details.

8.3.1.4 Com_IpduGroupStop

COM190:

Service name:	Com_IpduGroupStop	
Syntax:	`void Com_IpduGroupStop ( Com_PduGroupIdType IpduGroupId )`	
Service ID [hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different I-PDU groups. Non reentrant for the same I-PDU group.	
	`IpduGroupId`	Id of I-PDU group to be stopped
Parameters (out):	None	
Return value:	None	
Description:	Stops a preconfigured I-PDU group. For example, cyclic I-PDUs will be stopped after the call of `Com_IpduGroupStop()`. See Chapter 7.4.5 for details.	
Caveats:	A call to `COM_IpduGroupStop()` shall not be interrupted by another call to `COM_IpduGroupStop()` or a call to `COM_IpduGroupStart ()`. Note that this function is not only called by the COMM but also from other modules e.g. diagnosis.	
Configuration:	An I-PDU group must be configured before this call. See COM206 and COM341 for details.	

8.3.1.5 Com_DisableReceptionDM

COM192:

Service name:	Com_DisableReceptionDM	
Syntax:	`void Com_DisableReceptionDM ( Com_PduGroupIdType IpduGroupId )`	
Service ID [hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different I-PDU groups. Non reentrant for the same I-PDU group.	
Parameters (in):	`IpduGroupId`	Id of I-PDU group where reception DM shall be disabled.
Parameters (out):	None	
Return value:	None	
Description:	Disables the Reception Deadline Monitoring for the I-PDUs within the given I-PDU group. See Chapter 7.4.6.1.1 for details.	
Caveats:	--	
Configuration:	An I-PDU group must be configured before this call. See COM206 and COM341 for details.	

8.3.1.6 Com_EnableReceptionDM

COM193:

Service name:	Com_EnableReceptionDM	
Syntax:	`void Com_EnableReceptionDM ( Com_PduGroupIdType IpduGroupId )`	
Service ID [hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different I-PDU groups. Non reentrant for the same I-PDU group.	
Parameters (in):	`IpduGroupId`	Id of I-PDU group where reception DM shall be enabled.
Parameters (out):	None	
Return value:	None	
Description:	Enables the Reception Deadline Monitoring for the I-PDUs within the given I-PDU group. See Chapter 7.4.6.1.1 for details.	
Caveats:	--	
Configuration:	An I-PDU group must be configured before this call. See COM206 and COM341 for details.	

8.3.1.7 Com_GetStatus

COM194:

Service name:	Com_GetStatus	
Syntax:	`Com_StatusType Com_GetStatus ( void )`	
Service ID [hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	`None`	
Parameters (out):	`None`	
Return value:	`COM_UNINIT`	AUTOSAR COM is not initialized and not usable
	`COM_INIT`	AUTOSAR COM is initialized and usable
Description:	Returns the status of the AUTOSAR COM module	
Caveats:	--	
Configuration:	--	

8.3.1.8 Com_GetConfigurationId

COM375:

Service name:	Com_GetConfigurationId
Syntax:	`uint32 Com_GetConfigurationId ( void )`

Service ID [hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (out):	None	
Return value:	uint32	configured ConfigurationID, see COM394
Description:	Provides the unique identifier of the configuration.	
Caveats:	--	
Configuration:	The provided Identification shall be set during configuration process and can't be changed by COM	

8.3.1.9 Com_GetVersionInfo

COM426:

Service name:	Com_GetVersionInfo	
Syntax:	`void Com_GetVersionInfo ( Std_VersionInfoType *versioninfo )`	
Service ID [hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	None	
Parameters (out):	`versioninfo`	Pointer to where to store the version information of this module.
Return value:	None	
Description:	COM424: This service returns the version information of this module. The version information includes: Module ID Vendor ID Vendor specific version numbers (BSW00407). COM425: This function shall be pre compile time configurable On/Off by the configuration parameter: COM_VERSION_INFO_API Note: If source code for caller and called of this function is available, this function should be realized as a macro. The macro should be defined in the modules header file.	
Caveats:	--	
Configuration:	See COM026	

8.3.2 Communication services

8.3.2.1 Com_SendSignal

COM197:

Service name:	Com_SendSignal	
Syntax:	`uint8 Com_SendSignal ( Com_SignalIdType SignalId, Com_ApplicationDataRefType SignalDataPtr )`	
Service ID [hex]:	0x0A	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant for the same signal. Reentrant for different signals.	
Parameters (in):	SignalId	Id of signal to be sent.
	SignalDataPtr	Reference to the signal data to be transmitted.
Parameters (out):	None	
Return value:	E_OK	service has been accepted
	COM_SERVICE_ NOT_AVAILABLE	corresponding I-PDU group was stopped (or service failed due to development error)
Description:	The service `Com_SendSignal()` updates the signal object identified by SignalId with the signal referenced by the SignalDataPtr parameter. If the signal has the Triggered Transfer Property, the update is followed by immediate transmission of the I-PDU associated with the signal except when the signal is packed into an I-PDU with Periodic Transmission Mode; in this case, no transmission is initiated by the call to this service. If the signal has the Pending Transfer Property, no transmission is caused by the update.	
Caveats:	--	
Configuration:	--	

8.3.2.2 Com_ReceiveSignal

COM198:

Service name:	Com_ReceiveSignal	
Syntax:	`uint8 Com_ReceiveSignal ( Com_SignalIdType SignalId, Com_ApplicationDataRefType SignalDataPtr )`	
Service ID [hex]:	0x0B	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same signal. Reentrant for different signals.	
Parameters (in):	SignalId	Id of signal to be received.
Parameters (out):	SignalDataPtr	Reference to the signal data in which to store the received data.
Return value:	E_OK	service has been accepted

	COM_SERVICE_NOT_AVAILABLE	corresponding I-PDU group was stopped (or service failed due to development error)
Description:	The service `Com_ReceiveSignal()` updates the signal referenced by SignalDataPtr with the data in the signal object identified by SignalId.	
Caveats:	--	
Configuration:	--	

8.3.2.3 Com_UpdateShadowSignal

COM199:

Service name:	Com_UpdateShadowSignal	
Syntax:	`void Com_UpdateShadowSignal ( Com_SignalIdType SignalId, Com_ApplicationDataRefType SignalDataPtr )`	
Service ID[hex]:	0x0C	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same signal. Reentrant for different signals.	
Parameters (in):	SignalId	Id of signal (which belongs to a signal group) to be updated.
	SignalDataPtr	Reference to the signal data to be updated.
Parameters (out):	None	
Return value:	None	
Description:	The service `Com_UpdateShadowSignal()` updates a signal, which belongs to a signal group, with the data, referenced by SignalDataPtr. The update of the signal data is done in the shadow buffer, not in the I-PDU. To send out the shadow buffer, `Com_SendSignalGroup()` has to be called. For details see Chapter 7.5. Sign extension and byte swapping are performed as the signal is inserted into the shadow buffer.	
Caveats:	--	
Configuration:	A signal group must be configured before this call. See COM045 and Chapter 10.2.2.13 for details.	

8.3.2.4 Com_SendSignalGroup

COM200:

Service name:	Com_SendSignalGroup	
Syntax:	`uint8 Com_SendSignalGroup ( Com_SignalGroupIdType SignalGroupId )`	
Service ID [hex]:	0x0D	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant for the same signal group. Reentrant for different signal groups.	
Parameters (in):	SignalGroupId	Id of signal group to be send.

Parameters (out):	None	
Return value:	E_OK	service has been accepted
	COM_SERVICE_ NOT_AVAILABLE	corresponding I-PDU group was stopped (or service failed due to development error)
Description:	The service `Com_SendSignalGroup()` copies the content of the associated shadow buffer to the associated I-PDU. Prior to this call, all signals should be updated in the shadow buffer by the call of `Com_UpdateShadowSignal()`. For details see Chapter 7.5.	
Caveats:	--	
Configuration:	A signal group must be configured before this call. See COM045 and Chapter 10.2.2.13 for details.	

8.3.2.5 Com_ReceiveSignalGroup

COM201:

Service name:	Com_ReceiveSignalGroup	

Syntax:	`uint8 Com_ReceiveSignalGroup`
	`(`
	`  Com_SignalGroupIdType  SignalGroupId`
	`)`	

Service ID [hex]:	0x0E	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same signal group. Reentrant for different signal groups.	
Parameters (in):	SignalGroupId	Id of signal group to be received.
Parameters (out):	None	
Return value:	E_OK	service has been accepted
	COM_SERVICE_ NOT_AVAILABLE	corresponding I-PDU group was stopped (or service failed due to development error)
Description:	The service `Com_ReceiveSignalGroup()` copies the received signal group from the I-PDU to the shadow buffer. After this call, the signals from this signal group could be copied from the shadow buffer to the upper layer by a call of `Com_ReceiveShadowSignal()`. For details see Chapter 7.5.	
Caveats:	--	
Configuration:	A signal group must be configured before this call. See COM045 and Chapter 10.2.2.13 for details.	

8.3.2.6 Com_ReceiveShadowSignal

COM202:

Service name:	Com_ReceiveShadowSignal	

Syntax:	`void Com_ReceiveShadowSignal`
	`(`
	`  Com_SignalIdType  SignalId,`
	`  Com_ApplicationDataRefType  SignalDataPtr`
	`)`	

Service ID [hex]:	0x0F	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same signal. Reentrant for different signals.	

Parameters (in):	SignalId	Id of signal (which belongs to a signal group) to be received.
Parameters (out):	SignalDataPtr	Reference to the signal data in which to store the received data.
Return value:	None	
Description:	The service `Com_ReceiveShadowSignal()` updates the signal which belongs to a signal group and is referenced by SignalDataPtr with the data in the shadow buffer. The data in the shadow buffer should be updated before the call of `Com_ReceiveShadowSignal()` by a call of the service `Com_ReceiveSignalGroup()`. For details see Chapter 7.5.	
Caveats:	--	
Configuration:	--	

8.3.2.7 Com_InvalidateSignal

COM203:

Service name:	Com_InvalidateSignal	
Syntax:	`uint8 Com_InvalidateSignal ( Com_SignalIdType SignalId )`	
Service ID [hex]:	0x10	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant for the same signal. Reentrant for different signals.	
Parameters (in):	SignalId	Id of signal to be sent.
Parameters (out):	None	
Return value:	E_OK	service has been accepted
	COM_SERVICE_NOT_AVAILABLE	corresponding I-PDU group was stopped (or service failed due to development error)
Description:	Sender side AUTOSAR Software Component indicates via the RTE to AUTOSAR COM that it is not able to provide a valid value for the corresponding signal (e.g. sensor is faulty). After invaliding the actual signal data a `Com_SendSignal()` is performed internally, for details see COM097 and COM099.	
Caveats:	--	
Configuration:	For processing, a Data Invalid Value must have been configured (see COM338).	

8.3.2.8 Com_InvalidateShadowSignal

COM288:

Service name:	Com_InvalidateShadowSignal
Syntax:	`void Com_InvalidateShadowSignal ( Com_SignalIdType SignalId )`
Service ID [hex]:	0x16
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant for the same signal. Reentrant for different signals.

Parameters (in):	SignalId	Id of signal to be sent.
Parameters (out):	None	
	None	
Description:	Sender side AUTOSAR Software Component indicates via the RTE to AUTOSAR COM that it is not able to provide a valid value for the corresponding signal (e.g. sensor is faulty). For details see Chapter 7.4.3.6 and COM047. The RTE has to call this function for each signal inside a signal group. To send out the invalidated signals of a signal group `Com_SendSignalGroup()` must be called separately.	
Caveats:	--	
Configuration:	For processing, a Data Invalid Value must have been configured (see COM338).	

8.3.2.9 Com_TriggerIPDUSend

COM348:

Service name:	Com_TriggerIPDUSend	

Syntax:	`void Com_TriggerIPDUSend`
	`(`
	`  PduIdType ComTxPduId`
	`)`	

Service ID [hex]:	0x17	
Sync/Async:	Synchronous	
Reentrancy:	Non reentrant	
Parameters (in):	ComTxPduId	The I-PDU-ID if the I-PDU that shall be triggered for sending
Parameters (out):	None	
Return value:	None	
Description:	By a call to Com_TriggerIPDUSend the I-PDU with the given ID is triggered for transmission. COM388: When an I-PDU is sent out because of this API only the minimum delay time has to be taken into account. That is postpone transmissions if necessary and reset the minimum delay timer in case of transmissions. All other Transmission Mode related parameters like N-Times shall not be taken into account.	
Caveats:	Shall only be used from within an I-PDU callout	
Configuration:	--	

8.4 Call-back notifications

8.4.1 Com_TriggerTransmit

COM001:

Service name:	Com_TriggerTransmit	
Syntax:	`void Com_TriggerTransmit ( PduIdType ComTxPduId, uint8 *SduPtr )`	
Service ID [hex]:	0x13	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same PDU-ID. Reentrant for different PDU-ID.	
Parameters (in):	`ComTxPduId`	ID of AUTOSAR COM I-PDU that is requested to be transmitted by AUTOSAR COM.
Parameters (out):	`SduPtr`	Pointer to transmit buffer of L-PDU.
Return value:	None	
Description:	This function is called by the lower layer when an AUTOSAR COM I-PDU shall be transmitted. Within this function, AUTOSAR COM shall copy the contents of its I-PDU transmit buffer to the L-PDU buffer given by SduPtr. COM475: Com_TriggerTransmit is not interfered by the I-PDU minimum delay time (see COM181) and shall not reset the minimum delay timer. Use case: This function is used e.g. by the LIN Master for sending out a LIN frame. In this case, the trigger transmit can be initiated by the Master schedule table itself or a received LIN header. This function is also used by the FlexRay Interface for requesting PDUs to be sent in static part (synchronous to the FlexRay global time). Once the I-PDU has been successfully sent by the lower layer (PDU Router), the lower layer must call `Com_TxConfirmation()`.	
Caveats:	This function might be called in interrupt context.	
Configuration:	--	

8.4.2 Com_RxIndication

COM123:

Service name:	Com_RxIndication
Syntax:	`void Com_RxIndication ( PduIdType ComRxPduId, const uint8 *SduPtr )`
Service ID [hex]:	0x14
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant for the same PDU-ID. Reentrant for different PDU-ID.

Parameters (in):	`ComRxPduId`	ID of AUTOSAR COM I-PDU that has been received. Identifies the data that has been received. Range: 0..(maximum number of I-PDU IDs received by AUTOSAR COM) – 1
	`SduPtr`	Pointer to received SDU (payload buffer)
Parameters (out):	None	
Return value:	None	
Description:	This function is called by the lower layer after an I-PDU has been received.	
Caveats:	This function might be called in interrupt context. Therefore, data consistency must be ensured.	
Configuration:	--	

8.4.3 Com_TxConfirmation

COM124:

Service name:	Com_TxConfirmation	

Syntax:	`void Com_TxConfirmation`
	`(`
	`  PduIdType ComTxPduId`
	`)`	

Service ID [hex]:	0x15	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same PDU-ID. Reentrant for different PDU-ID.	
Parameters (in):	`ComTxPduId`	ID of AUTOSAR COM I-PDU that has been transmitted. Range: 0..(maximum number of I-PDU IDs transmitted by AUTOSAR COM) – 1
Parameters (out):	None	
Return value:	None	
Description:	This function is called by the lower layer after the PDU has been transmitted on the network. A confirmation that is received for an I-PDU that does not require a confirmation is silently discarded.	
Caveats:	This function might be called in interrupt context (e.g. from transmit interrupt).	
Configuration:	--	

8.5 Scheduled Functions

8.5.1 Com_MainFunctionRx

COM398:

Service name:	Com_MainFunctionRx
Service ID [hex]	0x18
Description:	This function shall perform the processing of the AUTOSAR COM receive processing that are not directly initiated by the calls from the RTE and PDU-R. A call to `Com_MainFunctionRx()` shall simply return if COM was not previously initialized with a call to `Com_Init()`.
Timing:	Cyclic
Pre condition:	None
Configuration:	See COM186.

8.5.2 Com_MainFunctionTx

COM399:

Service name:	Com_MainFunctionTx
Service ID [hex]	0x19
Description:	This function shall perform the processing of the AUTOSAR COM transmission activities that are not directly initiated by the calls from the RTE and PDU-R. A call to `Com_MainFunctionTx()` shall simply return if COM was not previously initialized with a call to `Com_Init()`.
Timing:	Cyclic
Pre condition:	--
Configuration:	See COM186

8.5.3 Com_MainFunctionRouteSignals

COM400:

Service name:	Com_MainFunctionRouteSignals
Service ID [hex]	0x1A
Description:	Calls the Signal Gateway part of COM to forward received signals to be routed. The insertion of this call is necessary for decoupling receive interrupts and Signal Gateway tasks. Caveat: The time between to consecutive calls (perhaps the related task/thread cycle) affects directly the Signal Gateway latency. A call to `Com_MainFunctionRouteSignals()` shall simply return if COM was not previously initialized with a call to `Com_Init()`.
Timing:	Cyclic
Pre condition:	--

Configuration:	A cyclic task/thread to call this function cyclical shall be configured. The cycle of this task/thread directly affects the latency of the Signal Gateway, see COM186.

8.6 Expected Interfaces

8.6.1 Mandatory Interfaces

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

API function	*Module*	*Description*
`PduR_ComTransmit`	PDU-R	Request the transmission of I-PDU.
`Dem_ReportErrorStatus`	DEM	Routine to report production relevant error events by event ID.

8.6.2 Optional Interfaces

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

API function	*Module*	*Description*	*Configuration parameter (description see Chapter 10)*
`Det_ReportError`	DET	Development error notification	COM_CONFIGURATION_USE_DET

8.6.3 Configurable interfaces

COM468:

Name:	Com Signal Notification
Syntax:	`void <CallbackRoutineName>(void)`
Reentrancy:	don't care
Parameters (in):	None
Parameters (out):	None
Return value:	None
Description:	This callback function corresponds to the notification classes 1 and 2 of [17].
Caveats:	A callback routine runs either on interrupt level or on task level. Thus, the OS restrictions of usage of system functions for interrupt service routines as well as for tasks apply.
Configuration:	For configuration of the callback function name, see COM247 and COM448.

COM491:

Name:	Com Signal Error	

Syntax:	`void <CallbackRoutineName>`
	`(`
	`uint8 error`
	`)`	

Reentrancy:	don't care	
Parameters (in):	`error`	`COM_TIMEOUT` – in case a timeout occurred
		`COM_SERVICE_NOT_AVAILABLE` – in case the corresponding I-PDU group was stopped, see also COM479
Parameters (out):	None	
Return value:	None	
Description:	This callback function corresponds to the notification classes 3 and 4 of [17].	
Caveats:	A callback routine runs either on interrupt level or on task level. Thus, the OS restrictions of usage of system functions for interrupt service routines as well as for tasks apply.	
Configuration:	For configuration of the callback function name, see COM247 and COM449.	

Note: The naming conventions for the RTE callback routines are defined in [13] in Chapter "Naming convention of callbackRoutineName".

Note: AUTOSAR COM uses no direct interface of RTE beside the callback functions.

9 Sequence diagrams

A sequence diagram of the underlying OSEK COM communication stack can be found in [17].

9.1 Interface between AUTOSAR COM and the lower layer (PDU Router)

The following chart shows the communication between AUTOSAR COM and the PDU Router.

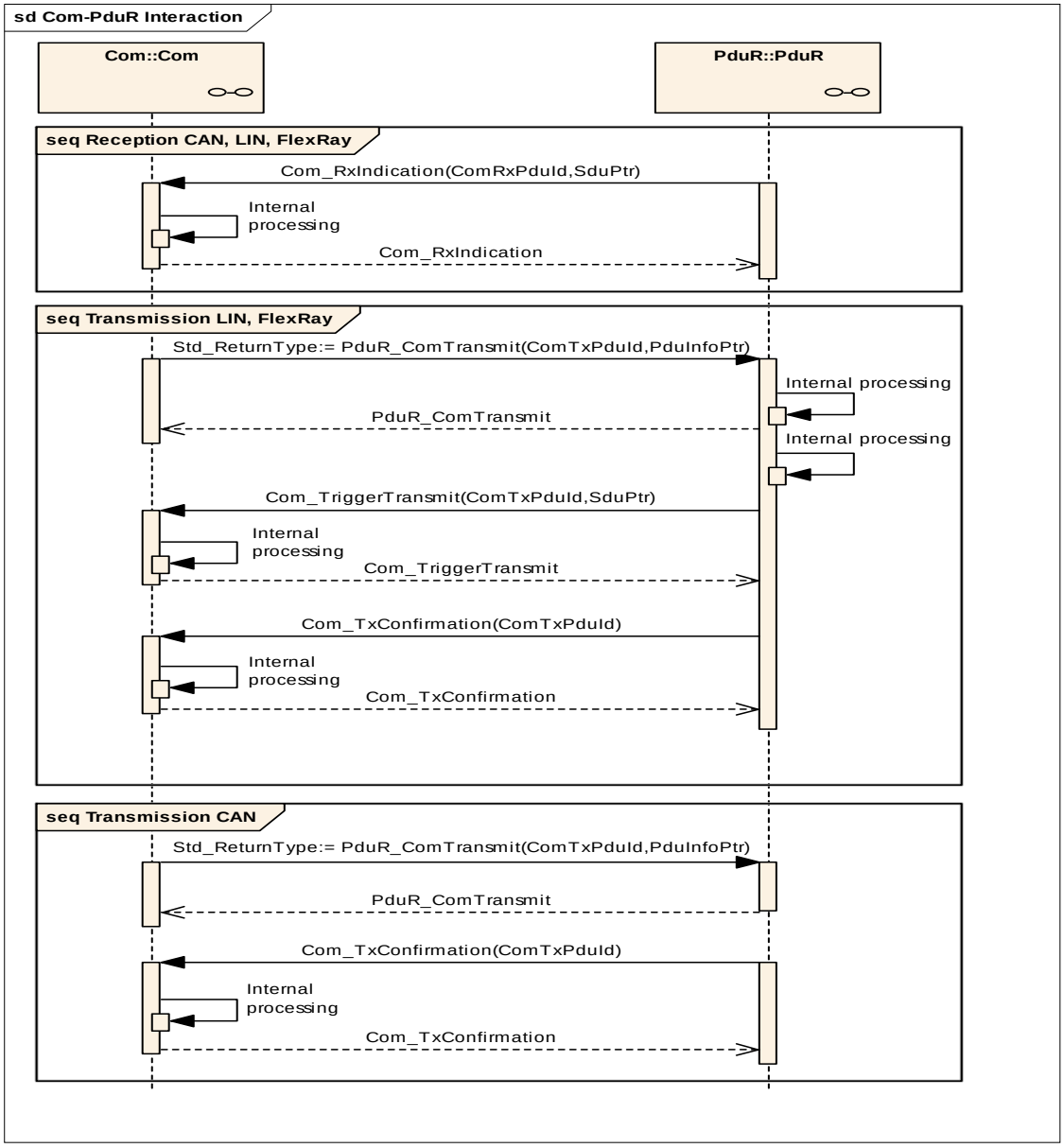

Figure 16: Interactions between AUTOSAR COM and the PDU router

9.2 Confirmation handling between PDUR, COM and RTE

The following chart shows the confirmation handling with respect to the two different IPDU-processing modes. (See also Chapter 7.4.5.3.)

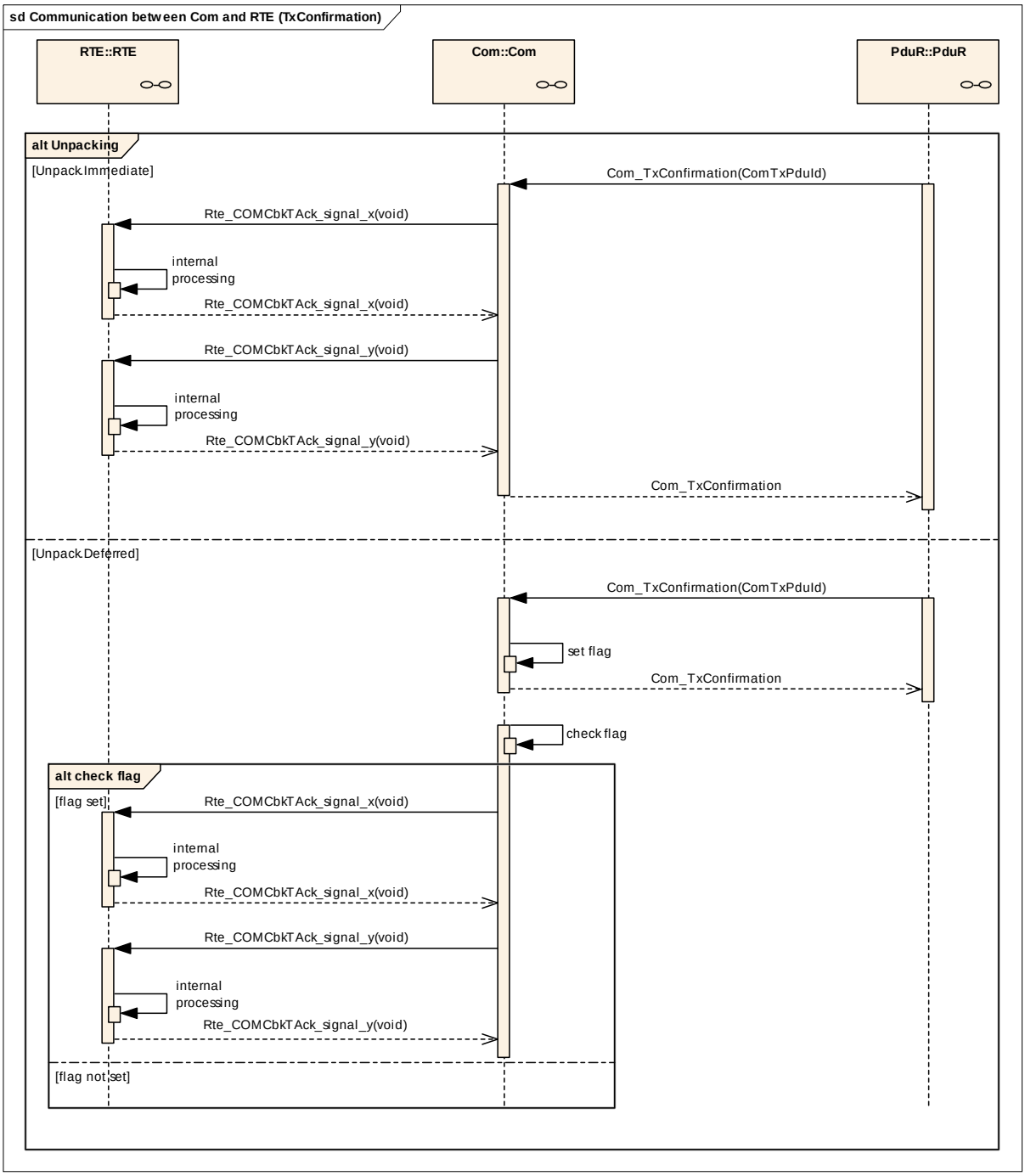

Figure 17: Confirmation handling between PDUR, COM and RTE

9.3 Indication handling between PDUR, COM and RTE

The following chart shows the indication handling with respect to the two different unpacking modes. (See also Chapter 7.4.5.3.)

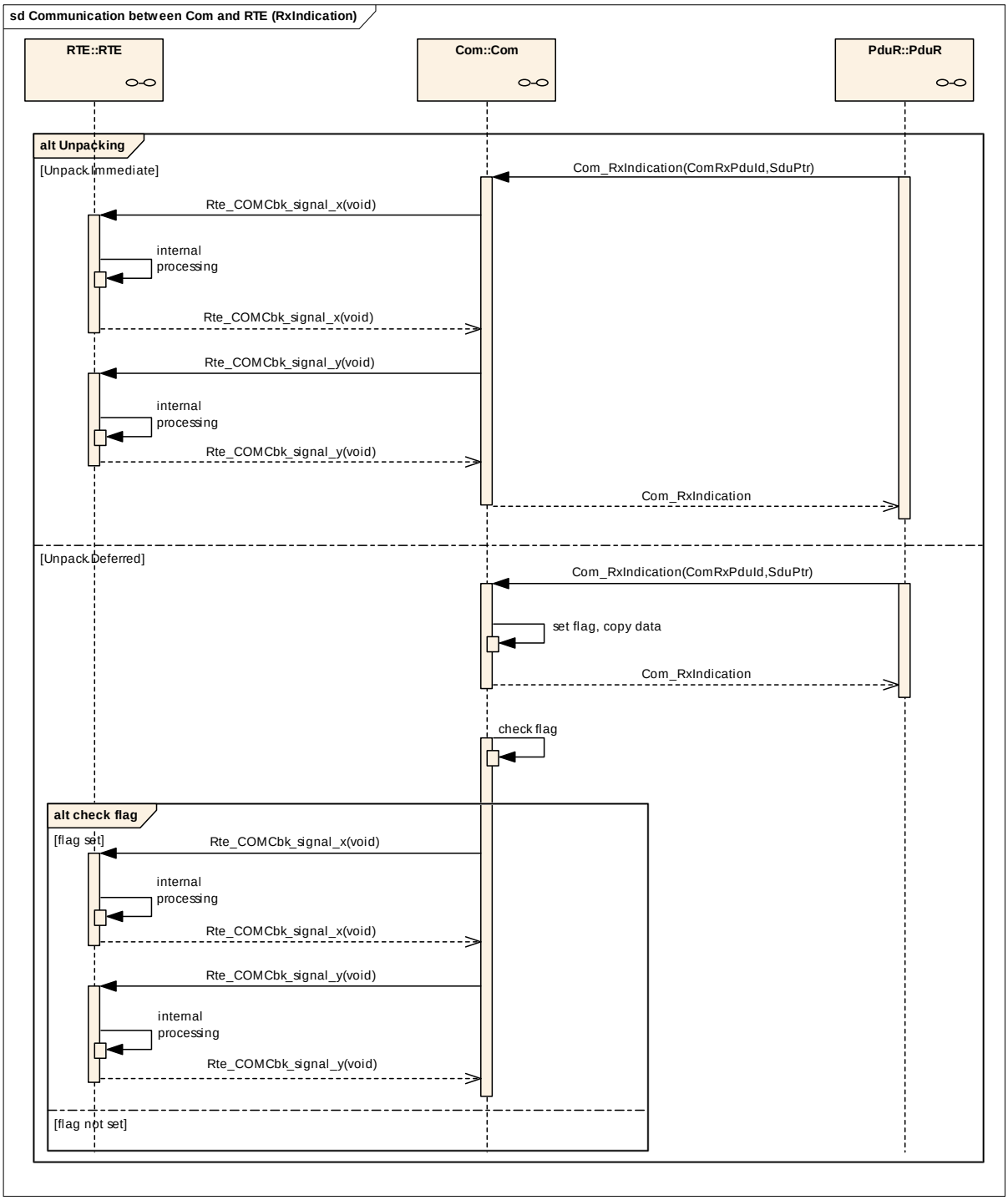

Figure 18: Indication handling between PDUR, COM and RTE

10 Configuration specification

10.1 How to read this chapter

In addition to this section, it is highly recommended to read the documents:

- AUTOSAR Layered Software Architecture [1]
- AUTOSAR ECU Configuration Specification [14]
  This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term ***configuration class*** (of a parameter) shall be used in order to refer to a specific configuration point in time.

COM006: The configuration parameters are based on [18]. All parameters have to be stored in an XML format.

10.1.2 Containers

Containers structure the set of configuration parameters. This means:
all configuration parameters are kept in containers.
(sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapter 7 and Chapter 8.

10.2.1 Variants

Currently three configuration variants for AUTOSAR COM are defined.

COM374: All configuration sets shall be identifiable by a unique identifier, see COM394.

10.2.1.1 Variant1 (Pre-compile Configuration)

Variant1 pre-compile configuration only supports pre-compile configurable parameters. Parameters below that are marked as Pre-compile configurable shall be configurable in a pre-compile manner, for example as #defines. A variant1-configurable module is most likely delivered as source code.

Remark: Even though the module is delivered as source code the implementation might use techniques similar to link time, i.e. table driven configuration.

10.2.1.2 Variant2 (Link-time Configuration)

Variant 2 includes mainly link-time and some pre-compile configurable parameters. All parameters defined below as link-time configurable shall be configurable at link time for example by linking a special configured parameter object file.
A variant2-configurable module is most likely delivered as object code.

10.2.1.3 Variant3 (Post-build Configuration)

Variant3 includes post-build-time, link-time and some pre-compile configurable parameters. All parameters defined below as post build configurable shall be configurable post build for example by flashing configuration data.
A variant3 configurable module is most likely delivered as object code.

10.2.2 Configuration of the AUTOSAR COM Layer

For an overview of the COM Configuration see Figure 19:

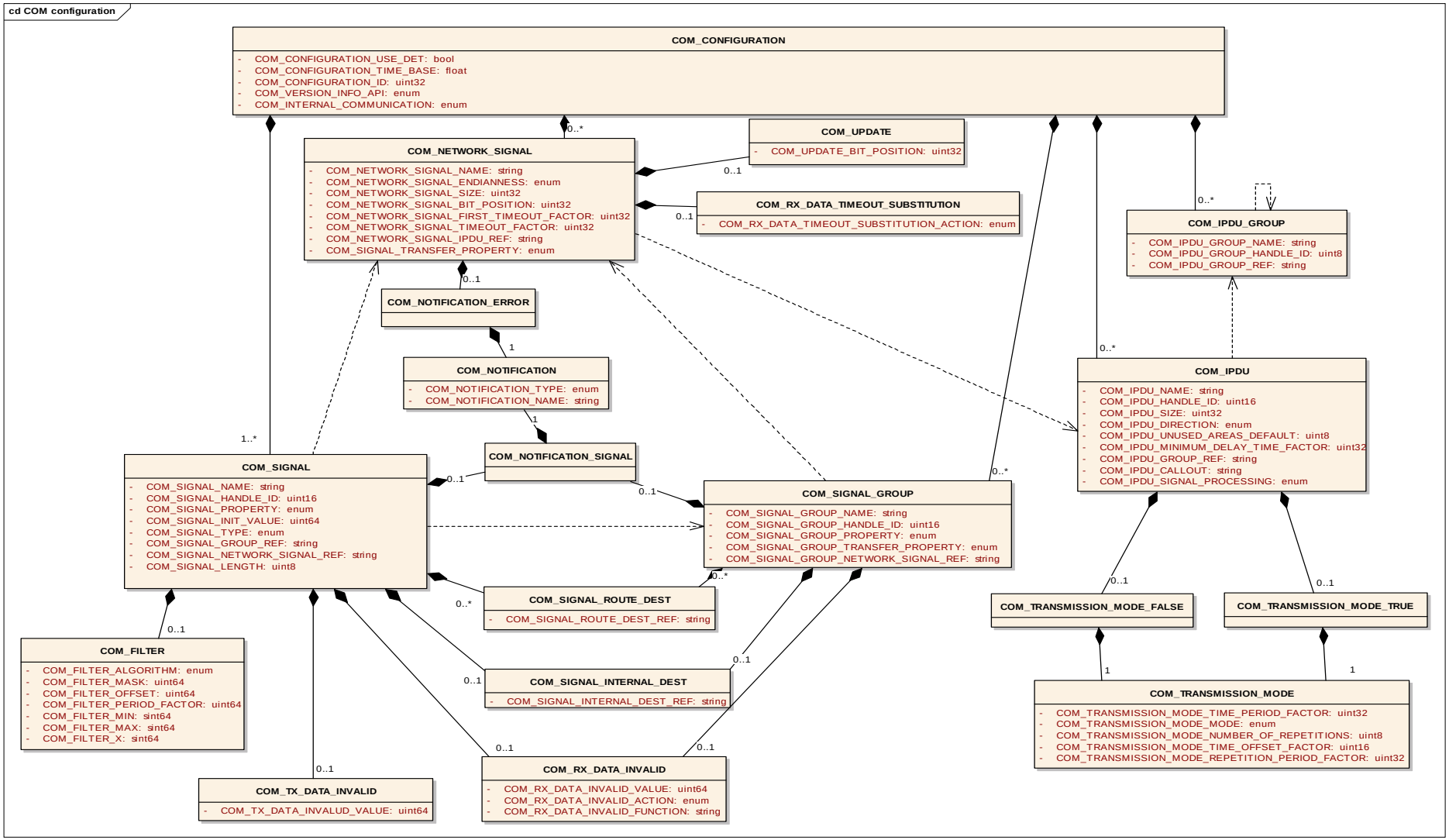

Figure 19: Com Configuration Overview

10.2.2.1 COM_CONFIGURATION

SWS Item	**COM337:**
Container Name	COM_CONFIGURATION
Description	This container contains the configuration parameters and sub containers of the COM Module.
Configuration Parameters	

Name	COM_CONFIGURATION_USE_DET		
Description	**COM141:** The error hook shall contain code to call the DET. If this parameter is configured `COM_DEV_ERROR_DETECT` shall be set to ON as output of the configuration tool (as input for the source code), see COM028.		
Type	boolean		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1, Variant2, Variant3
	Link time	--	--
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	COM_CONFIGURATION_TIME_BASE		
Description	**COM186:** The period between successive calls to the Main Functions (Rx, Tx, Routing) of AUTOSAR COM in seconds.		
Type	float		
Unit	seconds		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1, Variant2, Variant3
	Link time	--	--
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	COM_CONFIGURATION_ID		
Description	**COM394:** This ID is returned by a call to Com_**`GetConfigurationId()`**		
Type	uint32		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2
	Post Build	x	Variant3
Scope	Local		
Dependency	--		

Name	COM_VERSION_INFO_API
Description	**COM438:** Activate/Deactivate the version information API (see 8.3.1.9)
Type	Enum
Unit	--
Range	ON version information activated

	OFF version information deactivated		
Configuration Class	***Pre-compile***	x	Variant1, Variant2, Variant3
	Link time	--	--
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	COM_INTERNAL_COMMUNICATION		
Description	**COM490:** If this parameter is configured to ON AUTOSAR COM shall support internal communication (communication within an electronic control unit) as defined in [17]. If this parameter is configured to OFF no internal communication shall be supported. **Note:** This option shall allow to reduce possible COM overhead if no internal communication is used, for example if internal communication is handled by RTE. **Note:** There are considerations to remove the internal communication capabilities of AUTOSAR COM in later versions.		
Type	Enum		
Unit	--		
Range	ON internal communication is activated OFF internal communication is deactivated		
Configuration Class	***Pre-compile***	x	Variant1, Variant2, Variant3
	Link time	--	--
	Post Build	--	--
Scope	Local		
Dependency	--		

Included Containers

Container Name	***Multiplicity***	***Scope/Dependency***
COM_IPDU	0..*	**COM404:** See COM340. If there is no such container included no I-PDU is defined. In this case only internal communication is possible.
COM_IPDU_GROUP	0..*	**COM405:** See COM341 If there is no such container included then no I-PDU group is defined. In this case only internal communication is possible.
COM_NETWORK_SIGNAL	0..*	**COM406**: If there is no such container included no network signals are defined. In this case only internal communication is possible.
COM_SIGNAL	1..*	**COM407:** At least one signal container shall be present, see COM344.
COM_SIGNAL_GROUP	0..*	**COM408:** If there is no such container included no signal groups are defined.

10.2.2.2 COM_RX_DATA_INVALID

SWS Item	**COM338:**
Container Name	COM_RX_DATA_INVALID
Description	This container contains the configuration parameters of COM Data Invalid items This container is used for reception of signals.

Configuration Parameters

Name	COM_RX_DATA_INVALID_ACTION		
Description	**COM314:** This parameter defines the action performed upon reception of an invalid signal.		
Type	Enum		
Unit	--		
Range	NOTIFY, REPLACE		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	COM_RX_DATA_INVALID_INDICATION_FUNCTION		
Description	**COM315:** Name of the function which notifies the RTE about the reception of an invalidated signal. Only applicable if COM_DATA_INVALID_ACTION = Notify		
Type	String		
Unit	--		
Range			
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	COM_RX_DATA_INVALID_VALUE		
Description	**COM391:** When this value is received it is recognized as the invalid value and the appropriate invalid action (as specified by COM_DATA_INVALID_ACTION) is performed.		
Type	uint64		
Unit	--		
Range			
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	Local		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
--	--	--

10.2.2.3 COM_TX_DATA_INVALID

SWS Item	**COM447:**
Container Name	COM_TX_DATA_INVALID
Description	This container contains the configuration parameters of COM Data Invalid items. This container is used for transmission of signals.

Configuration Parameters

Name	COM_TX_DATA_INVALID_VALUE		
Description	**COM316:** The value will be set on sender side by a `Com_InvalidateSignal()` and `Com_InvalidateShadowSignal()` call.		
Type	uint64		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	Local		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
--	--	--

10.2.2.4 COM_RX_DATA_TIMEOUT_SUBSTITUTION

SWS Item	**COM488:**
Container Name	COM_RX_DATA_TIMEOUT_SUBSTITUTION
Description	This container contains the configuration parameters of COM Data substitution items if an Rx timeout violation occurs. This container is used only for reception of signals. In case of signal groups see also COM485.
Configuration Parameters	

Name	COM_RX_DATA_TIMEOUT_SUBSTITUTION_ACTION		
Description	**COM412:** This parameter defines the action performed upon a reception timeout violation. **COM470:** When this parameter is set to REPLACE, the replacement value used shall be the initial value of the signal.		
Type	Enum		
Unit	--		
Range	NONE REPLACE		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	Local		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
--	--	--

10.2.2.5 COM_FILTER

SWS Item	**COM339:**
Container Name	COM_FILTER
Description	This container contains the configuration parameters of COM Filters.
Configuration Parameters	

Name	COM_FILTER_ALGORITHM		
Description	**COM146:** The range of values is that specified in [17] Chapter 2.2.2, Reception Filtering.		
Type	Enum		
Unit	--		
Range	ALWAYS, NEVER, MASKED_NEW_EQUALS_X, MASKED_NEW_DIFFERS_X, MASKED_NEW_DIFFERS_MASKED_OLD, NEW_IS_WITHIN, NEW_IS_OUTSIDE, ONE_EVERY_N		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	COM_FILTER_MASK		
Description	**COM235:** The name of this attribute corresponds to the parameter name in [17] specification of Reception Filtering. Only the least significant 32 bits are significant.		
Type	uint64		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	COM_FILTER_PERIOD_FACTOR		
Description	**COM312:** The name of this attribute corresponds to the parameter name in [17] specification of Reception Filtering. Only the least significant 32 are significant.		
Type	uint64		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	COM_FILTER_OFFSET		
Description	**COM313:** The name of this attribute corresponds to the parameter name in [17] specification of Reception Filtering. Only the least significant 32 bits are significant.		
Type	uint64		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	COM_FILTER_X		
Description	**COM147:** The name of this attribute corresponds to the parameter name in [17] specification of Reception Filtering. Only the least significant 32 bits are significant.		
Type	sint64		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	COM_FILTER_MAX		
Description	**COM317:** The name of this attribute corresponds to the parameter name in [17] specification of Reception Filtering. Only the least significant 32 bits are significant.		
Type	sint64		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	COM_FILTER_MIN		
Description	**COM318:** The name of this attribute corresponds to the parameter name in [17] specification of Reception Filtering. Only the least significant 32 bits are significant.		
Type	sint64		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	local		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***

--	--	--

10.2.2.6 COM_IPDU

SWS Item	**COM340:**
Container Name	COM_IPDU
Description	This container contains the configuration parameters of COM I-PDUs.
Configuration Parameters	

Name	COM_IPDU_NAME		
Description	**COM174:** This attribute is used as the symbolic name of this I-PDU when communicating with the PDU-R. This parameter is only stored in the XML file, and must not be used within the implementation.		
Type	String		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	--	--
Scope	local		
Dependency	--		

Name	COM_IPDU_HANDLE_ID		
Description	**COM175:**The numerical value used as the ID of this I-PDU. The COM_IPDU_HANDLE_ID is required by the API calls to send and receive I-PDUs.		
Type	uint8		
Unit	--		
Range			
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	external, depends on configuration process		
Dependency	--		

Name	COM_IPDU_SIZE		
Description	**COM176:** The size of the I-PDU in bytes. The maximum size is limited by the underlying communication interface. 0-8 for CAN and LIN 0-254 for FlexRay		
Type	uint32		
Unit	Bytes		
Range	0 – 254		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2
	Post Build	x	Variant3
Scope	local		
Dependency	--		

Name	COM_IPDU_UNUSED_AREAS_DEFAULT
Description	**COM017:** AUTOSAR COM fills not used areas of an I-PDU with this bit-pattern. This attribute is mandatory to avoid undefined behavior. This byte-pattern will be repeated throughout the I-PDU.
Type	uint8

Unit	--		
Range	0 – 0xff		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	local		
Dependency	--		

Name	COM_IPDU_MINIMUM_DELAY_TIME_FACTOR		
Description	**COM181:** Minimum delay between successive transmissions of this I-PDU, independent of the Transmission Mode. Although this value is specified as a uint32 certain implementations may only use 16 bit counters and, therefore, some values may not always be possible. There is only one minimum delay time parameter for the I-PDU. This minimum delay time does not change with mode changes. Neither is the timer reset. This means that mode changes are not allowed to violate the minimum delay time. It is not possible to monitor the MINIMUMDELAYTIME for I-PDUs that are requested using the `Com_TriggerTransmit()` API. **COM471:** : A time of 0 shall mean that the monitoring of the delay time is not required.		
Type	uint32		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2
	Post Build	x	Variant3
Scope	local		
Dependency	--		

Name	COM_IPDU_CALLOUT		
Description	**COM387:** If there is a callout defined for this I-PDU this parameter contains the name of the callout function		
Type	String		
Unit	--		
Range			
Configuration Class	***Pre-compile***	x	Variant1, Variant2, Variant3
	Link time	--	--
	Post Build	--	--
Scope	local		
Dependency	--		

Name	COM_IPDU_GROUP_REF		
Description	**COM206:** Reference to the I-PDU group this I-PDU belongs to		
Type	String		
Unit	--		
Range			
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2
	Post Build	x	Variant3
Scope	local		

Dependency	--

Name	COM_IPDU_SIGNAL_PROCESSING		
Description	**COM119:** For the definition of the two modes Immediate and deferred, see COM298.		
Type	Enum		
Unit	--		
Range	IMMEDIATE DEFERED		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2
	Post Build	x	Variant3
Scope	local		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
COM_TRANSMISSION_MODE_TRUE	0..1	**COM233:** The referenced Transmission Mode object (see section10.2.2.14) that is used when the filtering state for this I-PDU evaluates to TRUE.
COM_TRANSMISSION_MODE_FALSE	0..1	**COM234:** The referenced Transmission Mode object (see section 10.2.2.14) that is used when the filtering state for this I-PDU evaluates to FALSE

10.2.2.7 COM_IPDU_GROUP

SWS Item	**COM341:**
Container Name	COM_IPDU_GROUP
Description	This container contains the configuration parameters of COM IPDU Groups.
Configuration Parameters	

Name	COM_IPDU_GROUP_NAME		
Description	**COM126:** This attribute used as the symbolic name of the I-PDU Group. This parameter is only stored in the XML file, and must not be used within the implementation.		
Type	String		
Unit	--		
Range			
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	--	--
Scope	local		
Dependency	--		

Name	COM_IPDU_GROUP_HANDLE_ID
Description	**COM184:** The numerical value used as the ID of this I-PDU Group. The COM_IPDUGROUP_HANDLE_ID is required by the API calls to start and stop I-PDU Groups.

	For the rational for the range see COM187.		
Type	uint8		
Unit	--		
Range	0..31		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	extern, depends on configuration process		
Dependency	--		

Name	COM_IPDU_GROUP_REF		
Description	**COM185:** If the I-PDU group belongs to an I-PDU group, this is the NAME of the I-PDU group it belongs to. If the I-PDU group does not belong to another I-PDU group, the string is empty.		
Type	String		
Unit	--		
Range			
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2
	Post Build	x	Variant3
Scope	local		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
--	--	--

10.2.2.8 COM_NETWORK_SIGNAL

SWS Item	**COM342:**
Container Name	COM_NETWORK_SIGNAL
Description	This container contains the configuration parameters of COM Network Signals.
Configuration Parameters	

Name	COM_NETWORK_SIGNAL_NAME		
Description	**COM156:** The symbolic name of the network signal. This parameter is only stored in the XML file, and must not be used within the implementation		
Type	String		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	--	--
Scope	local		
Dependency	--		

Name	COM_NETWORK_SIGNAL_ENDIANESS
Description	**COM157:** Defines the endianness of the network signal.

Type	Enum		
Unit	--		
Range	BIG_ENDIAN, LITTLE_ENDIAN, OPAQUE		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	COM_NETWORK_SIGNAL_SIZE		
Description	**COM158:** Size of the network signal in bits		
Type	uint32		
Unit	Bits		
Range	0..64		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	COM_NETWORK_SIGNAL_BIT_POSITION		
Description	**COM259:** Starting position in the I-PDU of the network signal. If the network signal refers to a signal inside a signal group, this parameter still refers to the position in the I-PDU and not in the shadow buffer.		
Type	uint32		
Unit	Bits		
Range	0 – 63		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2
	Post Build	x	Variant3
Scope	local		
Dependency	--		

Name	COM_NETWORK_SIGNAL_FIRST_TIMEOUT_FACTOR		
Description	**COM183:** Defines the first timeout period for the deadline monitoring. Details can be found in [17]. Note: See also COM263 for the configuration of the remaining timeout periods. Although this value is specified as a uint32 certain implementations may only use 16 bit counters and, therefore, some values may not always be possible.		
Type	uint32		
Unit	--		
Range			
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	local		
Dependency	--		

Name	COM_NETWORK_SIGNAL_TIMEOUT_FACTOR		
Description	**COM263:** Defines the timeout period for the deadline monitoring. Details can be found in [17]. Note: The period for the COM_NETWORK_SIGNAL_FIRST_TIMEOUT_FACTOR could differ from the COM_NETWORK_SIGNAL_TIMEOUT_FACTOR. Although this value is specified as a uint32 certain implementations may only use 16 bit counters and, therefore, some values may not always be possible. **COM264:** If deadline monitoring is used on a signal with an update bit this defines the timeout for deadline monitoring. (see COM292) **COM333:** If the timeout is configured to 0 than no timeout monitoring shall take place. In this case COM_NETWORK_SIGNAL_FIRST_TIMEOUT_FACTOR shall be ignored.		
Type	uint32		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	local		
Dependency	--		

Name	COM_NETWORK_SIGNAL_IPDU_REF		
Description	**COM161:** Reference to the I-PDU that contains this signal or signal group.		
Type	String		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2
	Post Build	x	Variant3
Scope	local		
Dependency	--		

Name	COM_SIGNAL_TRANSFER_PROPERTY		
Description	**COM232:** Derived from [18].		
Type	Enum		
Unit	--		
Range	TRIGGERED PENDING		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2
	Post Build	x	Variant3
Scope	local		
Dependency			

Included Containers		
Container Name	***Multiplicity***	***Scope/Dependency***
COM_UPDATE	0..1	COM_NETWORK_SIGNAL_UPDATE_REF **COM267:** If the signal has an update bit a COM_UPDATE

		container shall be included. **COM268:** If there is no COM_UPDATE container included, it implies the signal has not an update bit.
COM_RX_DATA_TIMEOUT_SUBSTITUTION	0..1	**COM411:** This container contains information for this signal if a reception timeout occurs. If there is no such container included then there is no data substitution for this signal shall take place.
COM_NOTIFICATION_ERROR	0..1	**COM451:** If there is no such container included then there is no notification for this signal configured.

10.2.2.9 COM_NOTIFICATION

SWS Item	**COM343:**
Container Name	COM_NOTIFICATION
Description	This container contains the configuration parameters of COM Notifications
Configuration Parameters	

Name	COM_NOTIFICATION_NAME		
Description	**COM247:** If COM_NOTIFICATION_TYPE is COM_CALLBACK then this is the name of a callback function. **COM248:** It shall be ensured, that the callback functions return type is void.		
Type	String		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	COM_NOTIFICATION_TYPE		
Description	**COM246:** Type of notification provided to RTE. Note that FLAG, ACTIVATETASK and SETEVENT, although present in OIL, have been omitted as they are not used in AUTOSAR RTE.		
Type	Enum		
Unit	--		
Range	NONE COM_CALLBACK		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant 3
	Post Build	--	--
Scope	Local		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope/Dependency***
--	--	--

10.2.2.10 COM_NOTIFICATION_SIGNAL

SWS Item	**COM448:**
Container Name	COM_NOTIFICATION_SIGNAL
Description	This container contains the configuration parameters of COM Notifications in the case a notification for a signal has to be done
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope/Dependency***
COM_NOTIFICATION	1	

10.2.2.11 COM_NOTIFICATION_ERROR

SWS Item	**COM449:**
Container Name	COM_NOTIFICATION_ERROR
Description	This container contains the configuration parameters of COM Notifications in the case a notification for an error has to be done.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope/Dependency***
COM_NOTIFICATION	1	

10.2.2.12 COM_SIGNAL

SWS Item	**COM344:**
Container Name	COM_SIGNAL
Description	This container contains the configuration parameters of COM Signals.
Configuration Parameters	

Name	COM_SIGNAL_NAME		
Description	**COM163:** The symbolic name of the signal. This name is also used as the handle name for the signal. This parameter is only stored in the XML file, and must not be used within the implementation.		
Type	String		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	--	--
Scope	local		
Dependency	--		

Name	COM_SIGNAL_HANDLE_ID		
Description	**COM165:** The numerical value used as the ID of this signal. The Com_SignalHandleId is required by the API calls to send and receive signals.		
Type	uint16		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	external, depends on configuration process		
Dependency			

Name	COM_SIGNAL_PROPERTY		
Description	**COM166:** Signal property definition according to section MESSAGEPROPERTY name, see [18] Chapter 3.2.10.1		
Type	Enum		
Unit	--		
Range	SEND_STATIC_INTERNAL SEND_STATIC_EXTERNAL RECEIVE_UNQUEUED_INTERNAL RECEIVE_UNQUEUED_EXTERNAL		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	local		
Dependency	--		

Name	COM_SIGNAL_INIT_VALUE		
Description	**COM170:** Initial value for this signal. Default value is 0. The lower n-bits of the configured uint64 shall be used as init-value for an n-bit sized signal type. **COM483:** If the signal is of type UINT[n], the uint64's least significant byte shall be assigned to the byte array's last byte. The second-least significant byte shall be assigned to the byte array's last but one byte, and so on.		
Type	uint64		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2
	Post Build	x	Variant3
Scope	local		
Dependency	IpduM		

Name	COM_SIGNAL_TYPE
Description	**COM127:** The AUTOSAR type of the signal. Whether or not the signal is signed or unsigned can be found by examining the value of this attribute. This type could also be used to reserved appropriate storage in AUTOSAR COM.
Type	Enum
Unit	--
Range	BOOLEAN,

	UINT8, UINT16, UINT32, SINT8, SINT16, SINT32, UINT8[n]		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	local		
Dependency	--		

Name	COM_SIGNAL_LENGTH		
Description	**COM437:** The COM_SIGNAL_LENGTH specifies the *n* of the type UINT8[n]. For other types it will be ignored.		
Type	uint8		
Unit	Byte		
Range	1 .. 8		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	local		
Dependency	--		

Name	COM_SIGNAL_GROUP_REF		
Description	**COM045:** If the signal belongs to a signal group, this is the name of the containing signal group. If the signal does not belong to any signal group, the parameter shall not appear.		
Type	String		
Unit	--		
Range			
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	COM_SIGNAL_NETWORK_SIGNAL_REF		
Description	**COM167:** If this signal is sent over a network this attribute is the name of matching network signal.		
Type	String		
Unit	--		
Range			
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	Local		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope/Dependency***

COM_FILTER	0..1	**COM169:** On receiver side: The name of the filter type as defined in the filter object in section 11.3.1. **COM275:** On sender side: Reference to a filter object which is used to determine the Transmission Mode Selector of the I-PDU the signal belongs to. If this attribute is omitted, the signal does not contribute to the evaluation of the Transmission Mode of the I-PDU the signal belongs to.
COM_NOTIFICATION_SIGNAL	0..1	**COM450:** If there is no such container included then there is no notification for this signal configured.
COM_RX_DATA_INVALID	0..1	**COM410:** This container contains the RX invalid information for this Com Signal If there is no such container included then there is no invalid information for this signal configured.
COM_TX_DATA_INVALID	0..1	**COM446:** This container contains the TX invalid information for this Com Signal. If there is no such container included then there is no invalid information for this signal configured.
COM_SIGNAL_ROUTE_DEST	0..*	**COM356:** If the signal is source of a signal gateway action, one or more Com_SignalRoutDest containers (one for each destination) shall be included. If there is no such container included then there no gating for this signal shall take place.
COM_SIGNAL_INTERNAL_DEST	0..1	**COM440:** If the signal is source of another internal signal (internal communication), one Com_SignalInternalDest container shall be included. If there is no such container included then there no gating for this signal shall take place.

10.2.2.13 COM_SIGNAL_GROUP

SWS Item	**COM345:**
Container Name	COM_SIGNAL_GROUP
Description	This container contains the configuration parameters of COM Signal-groups.
Configuration Parameters	

Name	COM_SIGNAL_GROUP_NAME		
Description	**COM044:** The symbolic name of the signal group. This name is also used as the handle name for the signal group. This parameter is only stored in the XML file, and must not be used within the implementation.		
Type	String		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	--	--
Scope	local		
Dependency	--		

Name	COM_SIGNAL_GROUP_HANDLE_ID		
Description	**COM149:** The numerical value used as the ID of this signal group. The Com_SignalGroupHandleId is required by the `Com_SendSignalGroup()` and `Com_ReceiveSignalGroup()` calls		
Type	uint8		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	external, depends on configuration process		
Dependency	--		

Name	COM_SIGNAL_GROUP_PROPERTY		
Description	**COM349:** Signal property definition according to section MESSAGEPROPERTY name in [18] Chapter 3.2.10.1.		
Type	Enum		
Unit	--		
Range	SEND_STATIC_INTERNAL, SEND_STATIC_EXTERNAL, RECEIVE_UNQUEUED_INTERNAL, RECEIVE_UNQUEUED_EXTERNAL		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	local		
Dependency	--		

Name	COM_SIGNAL_GROUP_TRANSFER_PROPERTY		
Description	**COM350:** Derived from [18]. For signal groups the Transfer Properties of the contained signals (COM_SIGNAL_TRANSFER_PROPERTY) shall be ignored.		
Type	Enum		
Unit	--		
Range	TRIGGERED, PENDING		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2
	Post Build	x	Variant3
Scope	Local		
Dependency	--		

Name	COM_SIGNAL_GROUP_NETWORK_SIGNAL_REF		
Description	**COM152:** Network signal that contains this signal group		
Type	String		
Unit	--		
Range			
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2, Variant3
	Post Build	--	--
Scope	Local		
Dependency	--		

Included Containers

Container Name	*Multiplicity*	*Scope/Dependency*
COM_NOTIFICATION_SIGNAL	0..1	**COM452:** If there is no such container included then there is no notification for this signal group configured.
COM_NOTIFICATION_ERROR	0..1	**COM453:** If there is no such container included then there is no notification for this signal group configured.
COM_SIGNAL_ROUTE_DEST	0..*	**COM415:** If the signal group is source of a signal gateway action, one or more COM_SIGNAL_ROUTE_DEST containers (one for each destination) shall be included. If there is no such container included then there no gating for this signal group shall take place.
COM_SIGNAL_INTERNAL_DEST	0..1	**COM441:** If the signal group is source of another internal signal (internal communication), one COM_SIGNAL_INTERNAL_DEST container shall be included. If there is no such container included then there no gating for this signal group shall take place.
COM_RX_DATA_TIMEOUT_SUBSTITUTION	0..1	**COM416:** This container contains information for this signal group if a reception timeout occurs. If there is no such container included then there is no data substitution for this signal group shall take place.
COM_RX_DATA_INVALID	0..1	**COM463:** This container contains the RX invalid information for this signal group. If there is no such container included then there is no invalid information for this signal configured.

10.2.2.14 COM_TRANSMISSION_MODE

SWS Item	**COM351:**
Container Name	COM_TRANSMISSION_MODE
Description	This container contains the configuration parameters of COM Transmission Modes
Configuration Parameters	

Name	COM_TRANSMISSION_MODE_TIME_PERIOD_FACTOR		
Description	**COM178:** Period of the repetition of cyclic transmissions. Although this value is specified as a uint32 certain implementations may only use 16 bit counters. Therefore the range is ECU and implementation dependant.		
Type	uint32		
Unit	--		
Range	--		
Configuration Class	*Pre-compile*	x	Variant1
	Link time	x	Variant2
	Post Build	x	Variant3
Scope	Local		
Dependency	--		

Name	COM_TRANSMISSION_MODE_MODE
Description	**COM137:** The available Transmission Modes described in [18] shall be extended by the additional Transmission Mode None.

	The Transmission Mode None shall not have any further sub-attributes in the COM_TRANSMISSION_MODE object. A detailed description of all Transmission Modes can be found in Chapter 7.4.3.1.		
Type	Enum		
Unit	--		
Range	DIRECT, PERIODIC, MIXED, NONE		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2
	Post Build	x	Variant3
Scope	Local		
Dependency	--		

Name	COM_TRANSMISSION_MODE_TIME_OFFSET_FACTOR		
Description	COM180: Time until first cyclic transmission of this I-PDU. Although this value is specified as a uint32 certain implementations may only use 16 bit counters Therefore the range is ECU and implementation dependant. COM_TRANSMISSION_MODE_TIME_OFFSET_FACTOR defines the time between `Com_IpduGroupStart()` and the first transmission of the cyclic part of this transmission request for this I-PDU.		
Type	uint32		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2
	Post Build	x	Variant3
Scope	Local		
Dependency	--		

Name	COM_TRANSMISSION_MODE_NUMBER_OF_REPETITIONS		
Description	**COM281:** Defines the number of repetitions for the Direct/N-Times Transmission Mode and the event driven part of Mixed Transmission Mode.		
Type	uint8		
Unit	--		
Range	0 – 255		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2
	Post Build	x	Variant3
Scope	--		
Dependency	--		

Name	COM_TRANSMISSION_MODE_REPETITION_PERIOD_FACTOR
Description	**COM282:** Period of the repetition of the n transmission for the Direct/N-Times Transmission Mode and the event driven part of the Mixed Transmission Mode. Although this value is specified as a uint32 certain implementations may only use 16 bit counters. Therefore the range is ECU and imple-

	mentation dependant.		
Type	uint32		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2
	Post Build	x	Variant3
Scope	local		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope/Dependency***
--	--	--

10.2.2.15 COM_TRANSMISSION_MODE_TRUE

SWS Item	**COM455:**
Container Name	COM_TRANSMISSION_MODE_TRUE
Description	This container contains the configuration parameters of COM Transmission Modes in the case the COM_FILTER evaluates to true
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope/Dependency***
COM_TRANSMISSION_MODE	1	

10.2.2.16 COM_TRANSMISSION_MODE_FALSE

SWS Item	**COM454:**
Container Name	COM_TRANSMISSION_MODE_FALSE
Description	This container contains the configuration parameters of COM Transmission Modes in the case the COM_FILTER evaluates to false
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope/Dependency***
COM_TRANSMISSION_MODE	1	

10.2.2.17 COM_UPDATE

SWS Item	**COM429:**
Container Name	COM_UPDATE
Description	This container contains the configuration parameters of COM Update items

Configuration Parameters

Name	COM_UPDATE_BIT_POSITION		
Description	**COM257:** Bit position of update bit inside I-PDU. **COM258:** If this signal is part of a signal group the signal must have no update bit. This setting must be consistently on sender and on receiver side. If this attribute is omitted then there is no update bit.		
Type	uint32		
Unit	Bits		
Range	0 – 63		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2
	Post Build	x	Variant3
Scope	local		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope/Dependency***
--	--	--

10.2.2.18 COM_SIGNAL_ROUTE_DEST

SWS Item	**COM382:**
Container Name	COM_SIGNAL_ROUTE_DEST
Description	Reference to a signal as destination of a signal gateway relation.
Configuration Parameters	

Name	COM_SIGNAL_ROUTE_DEST_REF		
Description	**COM355:** Name of a signal or signal group, which is destination of signal gateway relation.		
Type	String		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2
	Post Build	x	Variant3
Scope	local		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope/Dependency***
--	--	--

10.2.2.19 COM_SIGNAL_INTERNAL_DEST

SWS Item	**COM456:**
Container Name	COM_SIGNAL_INTERNAL_DEST
Description	Reference to a signal/signal group as internal destination
Configuration Parameters	

Name	COM_SIGNAL_INTERNAL_DEST_REF		
Description	**COM457:** Name of a signal or signal group, which is internal destination of the signal or signal group		
Type	String		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant1
	Link time	x	Variant2
	Post Build	x	Variant3
Scope	Local		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope/Dependency***
--	--	--

10.3 Published Information

Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information.

COM026: The following table specifies parameters that shall be published in the module's header file and also in the module's description file.

SWS Item		
Information elements		
Information element name	***Type / Range***	***Information element description***
`COM_VENDOR_ID`	`#define / uint8`	**COM208:** Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
`COM_MODULE_ID`	`#define / 0x32`	**COM417:** Module ID of this module from Module List
`COM_AR_MAJOR_VERSION`	`#define / uint8`	**COM418:** Major version number of AUTOSAR specification on which the appropriate implementation is based on.
`COM_AR_MINOR_VERSION`	`#define / uint8`	**COM419:** Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
`COM_AR_PATCH_VERSION`	`#define / uint8`	**COM420:** Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
`COM_SW_MAJOR_VERSION`	`#define /uint8`	**COM421:** Major version number of the vendor specific implementation of the module. The numbering is vendor

		specific.
COM_SW_MINOR_VERSION	#define / uint8	**COM422:** Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
COM_SW_PATCH_VERSION	#define / uint8	**COM423:** Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

10.4 Defines

Besides the configuration the following defines shall be implemented:

COM028: If COM_DEV_ERROR_DETECT is set to ON, the detection and reporting of development errors is enabled for the AUTOSAR COM module. For configuration of this parameter see COM141.

Note: This parameter shall be an output of the configuration tool. (Input for the source code)

10.5 Configuration rules

10.5.1 General rules

COM401: It is illegal for any two of the following parameters to have the same value:

- COM_RX_DATA_INVALID_FUNCTION
- COM_IPDU_NAME
- COM_IPDU_CALLOUT
- COM_NETWORK_SIGNAL_NAME
- COM_IPDU_GROUP_NAME
- COM_NOTIFICATION_NAME
- COM_SIGNAL_NAME
- COM_SIGNAL_GROUP_NAME

COM402: It is illegal for any of the following parameters not to be formulated according to C's identifier rules:

- COM_RX_DATA_INVALID_FUNCTION
- COM_IPDU_NAME
- COM_IPDU_CALLOUT
- COM_NETWORK_SIGNAL_NAME
- COM_IPDU_GROUP_NAME
- COM_NOTIFICATION_NAME
- COM_SIGNAL_NAME
- COM_SIGNAL_GROUP_NAME

10.5.2 Signal configuration

COM489: It shall be ensured, that the *Data Invalid Value* configured for the sender side is the same as configured for all receiver sides (see also COM097).

Note: The *Data Invalid Value* shall not be within the valid range of the signal. This can not be enforced by COM since knowledge about the application is needed.

More than one network signal can be packed into an I-PDU as long as the following packing rules are fulfilled:

COM101: No network signal shall span more than one I-PDU.

COM102: Network signals are not allowed to overlap each other.

COM105: Network signals representing a single signal and which are represented in I-PDUs as a multiple of 8-bits shall start at byte border only.

COM389: It shall be not allowed to configure a network signal of greater size than the COM signal.

Note: It is explicitly allowed that a network signal may have the size 0, see COM158.

COM443: A signal of type uint8[n] shall always be mapped to an n-bytes sized network signal.

COM474: The initial value of a signal shall always be within the possible range of the signal /network signal (including the Data Invalid Value).

Example: If a signal of data-type uint8 is mapped to a 6-bit sized network signal the possible range is 0..63. In this case it shall not be allowed to configure an init value of 64 or greater.

COM477: It shall not be allowed to configure a network signal which is not referred to by any signal.

10.5.3 Signal group configuration

Note: A signal group is packed within one network signal. COM105 shall not apply to network signals that represent signal groups.

COM216: All signals in a signal groups have to be placed in the same single I-PDU.

COM365: It shall not be allowed to configure signal groups for routing with data type differences between receive and transmit signal group.

COM482: For internally sent signal groups the initial value (of the shadow buffer) shall be the same on sender and receiver side.

COM485: In case of signal groups it shall not be allowed to configure a COM_RX_DATA_TIMEOUT_SUBSTITUTION container for the single signals. For signal groups this container shall only be configured for the whole signal group.

10.5.4 Transmission Mode configuration

COM319: It shall not be allowed to configure filter or TMS-conditions that uses floats. Floats are not allowed to be used in filter conditions. See [17] and COM132. Therefore floats are not allowed for conditions of TMS.

COM465: Every COM_TRANSMISSION_MODE_TRUE or COM_TRANSMISSION_MODE_FALSE that is a potential result of the configured/ calculated TMS must be configured. Within the COM_IPDU at least one of the containers COM_TRANSMISSION_MODE_FALSE or COM_TRANSMISSION_MODE_TRUE has to be included.

10.5.5 Signal Gateway configuration

COM384: The data type of a received and to be routed signal shall not differ.

COM385: In case of a signal to be routed by signal gateway has been configured for deadline monitoring at the receiving node, the related transmit signal(s) shall be configured to have update bit(s).

COM386: Optimization issue: In case of an I-PDU, containing signals to be routed completely via a transmit I-PDU by retention the signal order and the signals endianness (related use case: rate conversion), it can be configured to be handled en bloc.

10.5.6 Post Build Configuration

COM373: The post-build time configuration part (post-compile and post-link time) can only be updated when it is not in use

COM487: The whole post-build time configurable configuration shall be identifiable by a unique identifier.

11 Changes to Release 2.0

AUTOSAR COM 2.1 is a bug-fix release. The AUTOSAR COM passed the shortened review process and many clarifications and corrections compared to AUTOSAR COM 2.0 were introduced. This chapter lists all modified SWS items. Additionally referenced figures, tables, notes and so on were updated.

11.1 Deleted SWS Items

SWS Item	*Rationale*
COM160	network signal direction shall be derived from COM_SIGNAL_PROPERTY
COM177	IPDU direction shall be derived from COM_SIGNAL_PROPERTY

11.2 Replaced SWS Items

SWS Item	*Rationale*
COM139	can not be enforced by COM
COM306	turned into note (special case of COM428)
COM331	can not be enforced by COM turned into notes in the configuration rule chapter

11.3 Changed SWS Items

SWS Item	*Rationale*
COM001	clarified interaction of COM_TriggerTransmit and MDT
COM013	updated exclusion table according to 1:N mechanisms and callback routine name
COM053	removed signal error and invalid notifications for the single signals of a signal group
COM099	rephrased
COM100	rephrased
COM115	rephrased
COM117	rephrased
COM124	corrected parameter name
COM117	rephrased
COM146	restricted OSEK filter types
COM158	defined range of network-signal-size to 0..64
COM170	clarified configuration of initial value
COM178	changed timing parameter names to _factor
COM180	changed timing parameter names to _factor
COM181	changed timing parameter names to _factor
COM183	changed timing parameter names to _factor
COM190	clarified behavior of Com_IpduGroupStop
COM191	clarified behavior of Com_IpduGroupStart
COM197	changed return values according to RTE behavior
COM198	changed return values according to RTE behavior
COM199	removed error code on stopped I-PDU group for shadow-buffered APIs
COM200	changed return values according to RTE behavior
COM201	changed return values according to RTE behavior

SWS Item	Rationale
COM202	removed error code on stopped I-PDU group for shadow-buffered APIs
COM203	changed return values according to RTE behavior
COM220	updated include file structure according to General SRS
COM222	added parameter Initialize to COM_IpduGroupStart
COM224	clarification of Reception Deadline Monitoring
COM259	corrected range
COM263	changed timing parameter names to _factor
COM272	restricted OSEK filter types
COM282	changed timing parameter names to _factor
COM287	clarified requirement
COM288	removed error code on stopped I-PDU group for shadow-buffered APIs
COM302	rephrased
COM304	rephrased
COM305	excluded n==0 to define OSEK COM behavior for n==0
COM312	changed timing parameter names to _factor
COM320	rephrased
COM334	clarified behavior of stopped I-PDU groups
COM342	moved COM232, COM411 and COM451 to network signal container
COM344	moved COM232, COM411 and COM451 to network signal container
COM348	clarified interaction of MDT and Com_TriggerIPDUSend
COM374	clarification of post-build configuration
COM388	clarified interaction of Com_TriggerIPDUSend and MDT
COM393	defined what happens with the old_value of the filter in case of Rx-Timeout-Replacement
COM395	rephrased
COM396	rephrased
COM430	code file structure updated according to General SRS Template
COM439	restricted OSEK filter types
COM444	rephrased
COM459	replaced COM_E_STOPPED with COM_SERVICE_NOT_AVAILABLE
COM461	replaced COM_E_STOPPED with COM_SERVICE_NOT_AVAILABLE

11.4 Added SWS Items

SWS Item	Rationale
COM462	clarification of behavior of signals within a signal group
COM463	corrected configuration of signal groups
COM464	restricted endianness within a signal group
COM465	added configuration rule for Transmission Modes
COM466	defined internal sequence of COM_MainFunctionRouteSignals
COM467	redefined N-Times Transmission Mode with n==0 to OSEK COM behavior
COM468	defined callback routine naming conventions
COM469	clarification of minimum delay timer
COM470	defined replacement value of RX-timeout substitution to initial value
COM471	clarification of minimum delay timer
COM472	restricted opaque date to byte arrays
COM473	defined endianness for opaque types
COM474	added configuration rule for the initial value
COM475	clarified interaction of COM_TriggerTransmit and MDT
COM476	clarified redundant start/stops of I-PDU groups
COM477	added configuration rule for network signals
COM478	clarification of Com_MainFunctionTx behavior
COM479	changed error/confirmation handling of stopped I-PDU groups
COM480	restricted OSEK filter types

SWS Item	*Rationale*
COM481	simplification of Transmission Deadline Monitoring
COM482	configuration rule for internally sent signal groups
COM483	clarified interpretation of initial value in case of UINT[n] signal type
COM484	defined initialization for shadow buffers of signal groups
COM485	restricted usage of COM_RX_DATA_TIMEOUT_SUBSTITUTION for signal groups
COM486	transformed note below COM224 into new requirement.
COM487	requirement ID COM374 was used twice, changed to second occurrence to COM487
COM488	requirement ID COM394 was used twice, changed to second occurrence to COM488
COM489	transformed first note in Chapter 10.5.2 to requirement COM489
COM490	configurability of Internal communication support
COM491	defined callback routine naming conventions

12 Appendix A

In the following use cases with different Transmission Modes and the necessary configuration for these are shown.
For the legend of the pictures see Chapter 7.4.3.6.

Use Case A1 shows an I-PDU which is send out cyclically with a cycle time t_c. This I-PDU consists of signals which all have the Pending Transfer Property. It is configured that the send out takes place when the TMS evaluates to TRUE.

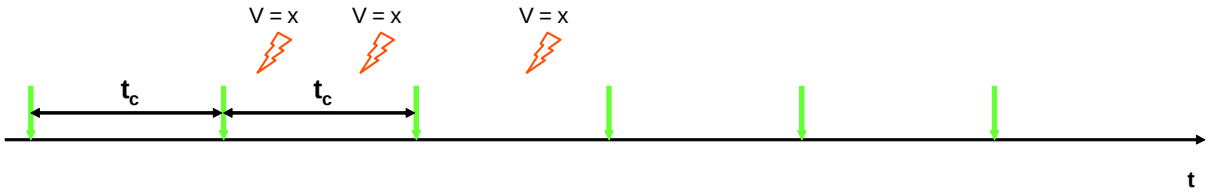

Figure 20: Use Case A1, TM periodic (without TMS switch (see Chapter 7.4.3.3))

Relevant configuration items for the I-PDU transmission	
Object SIGNAL	
TRANSFERPROPERTY	Pending or Triggered (Triggered has no influence)
Object COM_FILTER	
FILTERALGORITHM	F_ALWAYS
Object I-PDU	
PROPERTY	Sent
Object TRANSMISSION MODE on filter TRUE	
TIMEPERIOD	t_c
MODE	periodic
NUMBER_OF_REPETITIONS	N/a
REPETITION_PERIOD	N/a
Object TRANSMISSION MODE on filter FALSE	
	N/a

Because of the configuration of the parameter FILTERALGORITHM (F_ALWAYS) of the COM_FILTER, there is no need to configure a Transmission Mode for the case that the TMS evaluates to FALSE.
It does not make any difference in the behavior whether the FILTERALGORITHM parameter of the COM_FILTER is defined in the configuration for all the signals within the I-PDU with F_ALWAYS or if the COM_FILTER is not defined (shall not contribute to the evaluation of the TMS), see COM255.

Use Case A2 shows an I-PDU which is sent out three times whenever a value is given by the upper (`Com_SendSignal()` or `Com_SendSignalGroup()`). The time between two send outs is t_d. This I-PDU consists of signals which all have the Triggered Transfer Property. It is configured that the send out takes place when the TMS evaluates to TRUE.

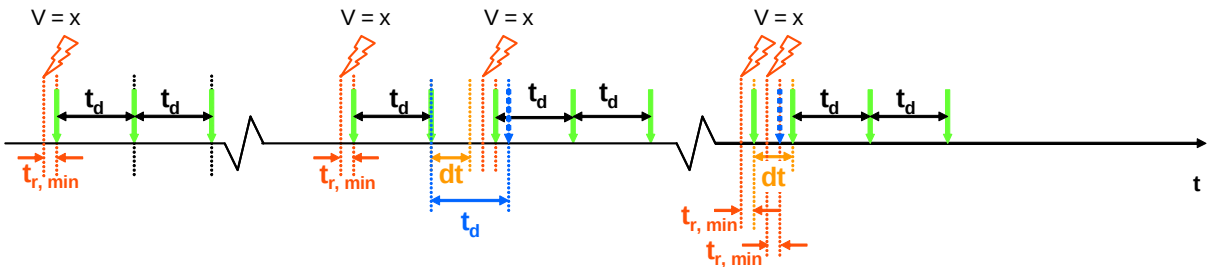

Figure 21: Use Case A2, TM Direct/N-Times, here n = 3 (without TMS switch)

Relevant configuration items for the I-PDU transmission	
Object SIGNAL	
TRANSFERPROPERTY	Triggered
Object COM_FILTER	
FILTERALGORITHM	F_ALWAYS
Object I-PDU	
PROPERTY	Sent
Object TRANSMISSION MODE on filter TRUE	
TIMEPERIOD	N/a
MODE	Direct/N-Times
NUMBER_OF_REPETITIONS	3
REPETITION_PERIOD	t_d
Object TRANSMISSION MODE on filter FALSE	
	N/a

If there is a new send request by the upper layer before the last three sent outs have taken place, the new sent out is started and the rest of the last one is discarded.

Use Case A3 shows an I-PDU which is send out cyclically with a cycle time t_{c1} if value v = a (TMS evaluates to TRUE) and with a cycle time t_{c2} if value v = b (TMS evaluates to FALSE). The I-PDU consists of signals which all have the Pending Transfer Property.

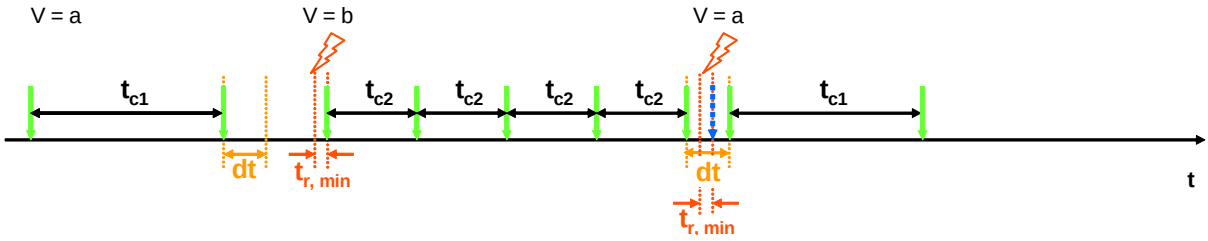

Figure 22: Use Case A3, TM periodic + periodic (with TMS switch)

Relevant configuration items for the I-PDU transmission	
Object SIGNAL	
TRANSFERPROPERTY	Pending or Triggered (Triggered has no influence)
Object COM_FILTER	
FILTERALGORITHM	all except F_ALWAYS and F_Never
Object I-PDU	
PROPERTY	Sent
Object TRANSMISSION MODE on filter TRUE	
TIMEPERIOD	t_{c1}
MODE	periodic
NUMBER_OF_REPETITIONS	N/a
REPETITION_PERIOD	N/a
Object TRANSMISSION MODE on filter FALSE	
TIMEPERIOD	t_{c2}
MODE	periodic
NUMBER_OF_REPETITIONS	N/a
REPETITION_PERIOD	N/a

Because of the TMS switch caused by the new value v = b, the new cycle is started immediately and the new value is sent out. But nevertheless the minimum delay time dt has to be taken into account.
For the parameter FILTERALGORITHM of the configuration object COM_FILTER every in OSEK COM defined item can be used except F_ALWAYS and F_NEVER. These are:

- F_Always
- F_Never
- F_MaskedNewEqualsX
- F_MaskedNewDiffersX
- F_MaskedNewDiffersMaskedOld
- F_NewIsWithin
- F_NewIsOutside
- F_OneEveryN

If the FILTERALGORITHM F_OneEveryN is used not the value of the signal itself has an influence to the TMS but the number of send requests by the upper layer.

Use Case A4 shows an I-PDU which is send out cyclically with a cycle time t_c if value v = a (TMS evaluates to TRUE) and if value v = b (TMS evaluates to FALSE) it is sent out three times whenever the value is given by the upper layer. The time between two send outs is t_d. The I-PDU consists of signals which all have the Triggered Transfer Property.

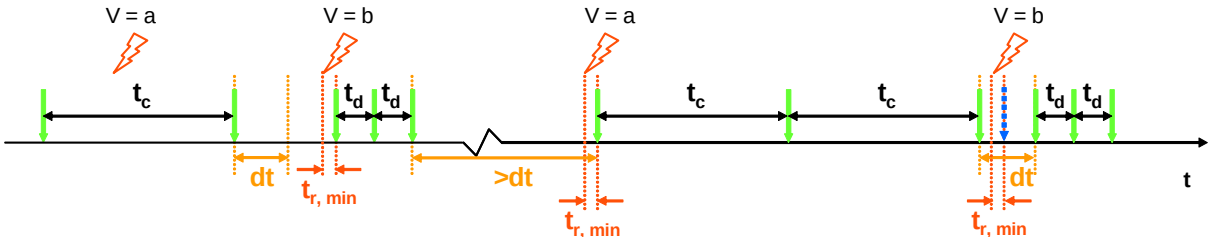

Figure 23: Use Case 4, TM periodic + Direct/N-Times, here n = 3 (with TMS switch)

Relevant configuration items for the I-PDU transmission	
Object SIGNAL	
TRANSFERPROPERTY	Triggered
Object COM_FILTER	
FILTERALGORITHM	all except F_ALWAYS and F_Never
Object I-PDU	
PROPERTY	Sent
Object TRANSMISSION MODE on filter TRUE	
TIMEPERIOD	t_c
MODE	periodic
NUMBER_OF_REPETITIONS	N/a
REPETITION_PERIOD	N/a
Object TRANSMISSION MODE on filter FALSE	
TIMEPERIOD	N/a
MODE	Direct/N-Times
NUMBER_OF_REPETITIONS	3
REPETITION_PERIOD	t_d

After the switch from Direct/N-Times to periodic the cycle is started immediately and the new value a is sent out (with respect of the minimum delay time dt).

Use Case A5 shows an I-PDU which is send out cyclically with a cycle time t_c and if the value (the same or a new one) is given by the upper layer it is also sent out directly three times. The time between two of these three send outs is always t_d. The I-PDU consists of signals which all have the Triggered Transfer Property.

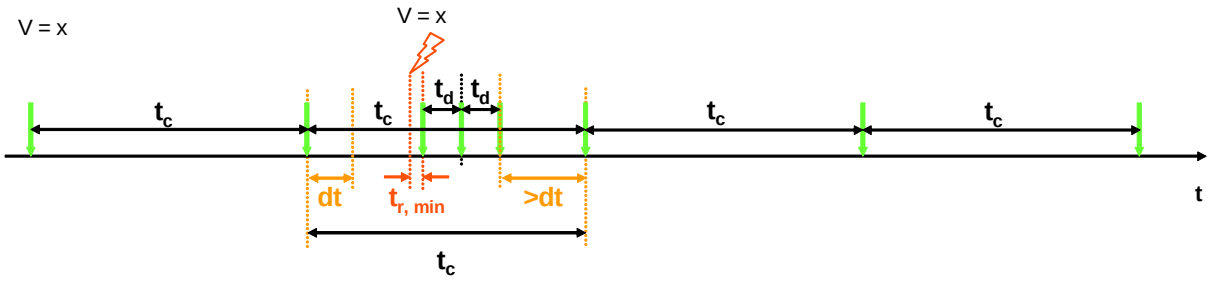

Figure 24: Use Case 5a, TM Mixed, here n = 3 (without TMS switch)

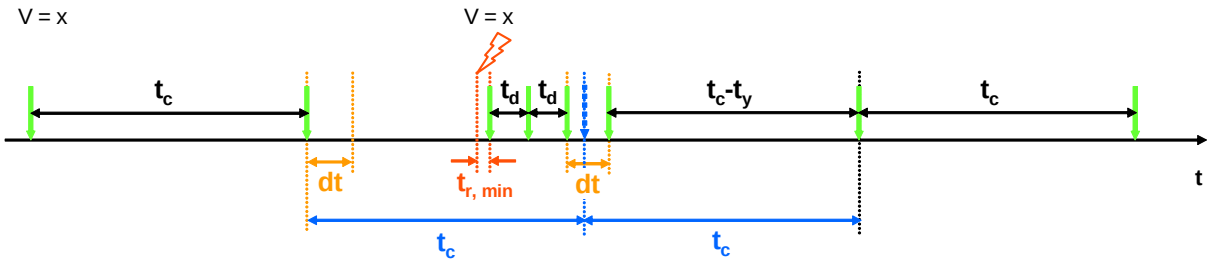

Figure 25: Use Case 5b, TM Mixed, here n = 3 (without TMS switch), no phase shift

Relevant configuration items for the I-PDU transmission	
Object SIGNAL	
TRANSFERPROPERTY	Triggered
Object COM_FILTER	
FILTERALGORITHM	F_ALWAYS
Object I-PDU	
PROPERTY	Sent
Object TRANSMISSION MODE on filter TRUE	
TIMEPERIOD	t_c
MODE	Mixed
NUMBER_OF_REPETITIONS	3
REPETITION_PERIOD	t_d
Object TRANSMISSION MODE on filter FALSE	
	N/a

If the next sent out caused by the periodic part of the Mixed Transmission Mode should take place within the timeout dt (minimum delay time) after a sent out of the Direct/N-Times part, this sent out is delayed until the minimum delay time is elapsed. But after that the next time period of the periodic part is shortened so that there is only an intermediate phase shift of the periodic part but no continuous one.

Use Case A6 shows an I-PDU which is send out cyclically with a cycle time t_{c2} if value v = b (TMS evaluates to FALSE). If value v = a (TMS evaluates to TRUE) it is sent out cyclically with a cycle time t_{c1} and whenever the value v = a is given by the upper layer it is also sent out directly three times. The time between two of these three send outs is always $t_{d.}$ The I-PDU consists of signals which all have the Triggered Transfer Property.

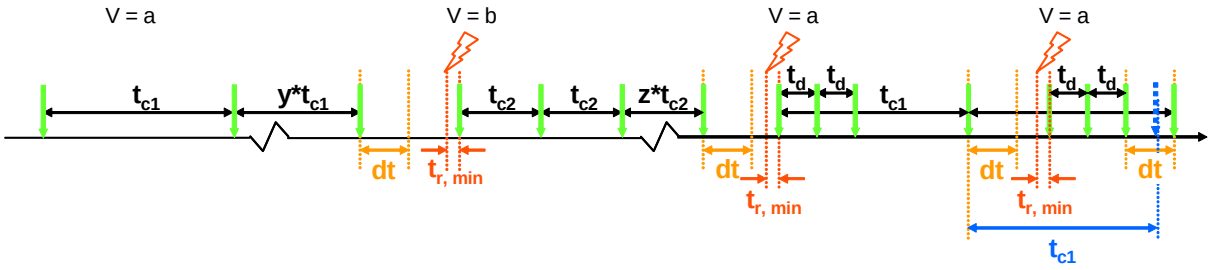

Figure 26: Use Case 6, TM Mixed, here n= 3 + periodic (with TMS switch)

Relevant configuration items for the I-PDU transmission	
Object SIGNAL	
TRANSFERPROPERTY	Triggered
Object COM_FILTER	
FILTERALGORITHM	all except F_ALWAYS and F_Never
Object I-PDU	
PROPERTY	Sent
Object TRANSMISSION MODE on filter TRUE	
TIMEPERIOD	t_{c1}
MODE	Mixed
NUMBER_OF_REPETITIONS	3
REPETITION_PERIOD	t_d
Object TRANSMISSION MODE on filter FALSE	
TIMEPERIOD	t_{c2}
MODE	periodic
NUMBER_OF_REPETITIONS	N/a
REPETITION_PERIOD	N/a

A usage of this in practice is for example the signal of the button that controls the window-lift motor. If the button is not pressed there is a long cycle time t_{c1} with this information. If it is pressed this information is distributed with a short cycle time t_{c2}. If the button is released again, this information is immediately distributed three times with t_d and after that again the long cycle time is used.