MICROSAR Classic COM

Technical Reference

CFG5

Version 14.01.00

Authors	visdbi, visssa, visms, vishho, visbms, visgut, visbbk, visssg, viscpe, alefarth, oalka, vadnra
Status	Released

Document Information

History

Author	Date	Version	Remarks
visssa visdbi	2012-02-21	1.00.00	> Adapted to Cfg5
visdbi	2012-07-19	2.00.00	> Adapted to AUTOSAR 4.0.3 (ESCAN00058304)
visdbi	2012-12-11	2.01.00	> Added new feature descriptions > Invalidation Mechanism (ESCAN00061795) > Signal Reception Filtering (ESCAN00061805) > Signal Status Information (ESCAN00061799)
visms	2013-04-04	2.02.00	> AR4-325: Post-Build Loadable (ESCAN00064360)
visdbi	2013-04-12	2.02.00	> Added new feature Signal Gateway (ESCAN00064081) Adapted > Critical Sections (ESCAN00065327)
visdbi	2013-06-20	3.00.00	> Changed prototype description of Callout Functions (ESCAN00068096)
visdbi	2013-09-03	3.00.00	> Transmission Deadline Monitoring (ESCAN00070185) > Clarified transmission context of signals and signal groups (ESCAN00070200) > Updated limitations of Callout Functions (ESCAN00067146) > Support (ESCAN00070186) > Added support for 0-Bit signals (ESCAN00069728)
vishho	2013-09-23	3.00.00	> Added following API descriptions (ESCAN00070449): > Com_TpTxConfirmation > Com_CopyTxData > Com_TpRxIndication > Com_StartOfReception > Com_CopyRxData > Com_SendDynSignal > Com_ReceiveDynSignal

vishho	2013-10-02	3.00.00	> Added Chapter Large I-PDUs and Dynamic length signals (ESCAN00070449)
vishho	2014-03-25	3.01.00	> Updated Com_ StartOfReception API according to ASR4.1.2 (ESCAN00071922)
vishho	2014-04-16	3.01.00	> ESCAN00075083: AR4-710: Support IPDUGroup relevant API like as ASR3 API > Com_IpduGroupStart > Com_IpduGroupStop > Com_EnableReceptionDM > Com_DisableReceptionDM
vishho	2014-04-25	3.01.00	> Added Chapter: Autosar 3 based I-Pdu Group handling
visbms	2014-06-18	3.01.00	> AR4-601: Support RfC #59936 (Request Handling)
vishho	2014-06-27	3.01.00	> Added chapter 2.1.5 Signal Processing
vishho	2014-06-27	3.01.01	> ESCAN00076544: The type of the parameter "Result" of the APIs Com_TpRxIndication and Com_TpTxConfirmation mismatches between TechRef and source
vishho	2014-11-05	3.02.00	> ESCAN00079371: AR4-698: Post-Build Selectable (Identity Manager)
vishho	2015-05-11	3.03.00	> ESCAN00082938: FEAT-1047: Gateway Rx signal timeout handling without update bits [AR4-894]
vishho	2015-05-20	3.03.01	> ESCAN00083061: The figure on page 11 is labelled twice.
visgut	2015-07-15	4.01.00	> ESCAN00084005: FEAT-77: COM Based Transformer for CONC_601_SenderReceiverSerialization incl. E2EXf [AR4-829]
visgut	2015-12-07	5.00.00	> ESCAN00085557: FEAT-1436 Gateway: Optimization for COM. > ESCAN00087097: FEAT-1442 Gateway: Routing timeout behaviour
visms	2016-02-25	5.01.00	> ESCAN00088555: FEAT-1631: Trigger Transmit API with SduLength In/Out according to ASR4.2.2
visms visgut	2016-06-05	5.02.00	> ESCAN00090302: FEAT-1688: SafeBSW Step 4

visbbk	2016-07-13	5.03.00	> ESCAN00090995: FEAT-1818: ComEnableMDTForCyclicTransmission
visgut	2016-07-13	5.03.00	> ESCAN00091008: FEAT-1640: > Notification caching and ISR Lock Thresholds
visgut	2016-07-26	5.03.00	> ESCAN00091171: Missing description of Com_Cot.h
visgut	2016-08-10	5.03.01	> ESCAN00091398: Update API description.
visbbk	2016-09-28	5.04.00	> ESCAN00092101: FEAT-2002: Support 64 Bit Signal Types for COM according to ASR 4.2.2
visgut	2016-10-26	5.04.00	> ESCAN00092539: FEAT-1890: Extension of gateway description routing with shifted copy and update bit support.
vishho	2016-11-07	5.04.00	> ESCAN00092684: Tx Timeout for Cyclic Messages misses in "Not Supported AUTOSAR Standard Conform Features" chapter
visbbk	2016-12-09	5.05.00	> FEATC-793: FEAT-2231 Support ComTxRepetitionCnt according to ASR 4.2.2
visgut	2017-01-17	5.05.00	> FEATC-797: FEAT-2333: Support Tx Timeout for cyclic messages
visgut	2017-04-10	6.00.00	> STORYC-18: Support IPDUs without IPDU Group assignment in COM
visbbk	2017-04-14	6.00.00	> STORYC-164: Implement MASKED_NEW_EQUALS_X and MASKED_NEW_DIFFERS_X for Signal Group Arrays in Com
visbbk	2017-05-11	6.01.00	> STORYC-694: Implement MASKED_NEW_EQUALS_X and MASKED_NEW_DIFFERS_X for Signal Group Arrays in Com
visbbk	2017-05-17	6.02.00	> STORYC-1205: Rework of COM
visgut	2017-07-05	6.03.00	> STORYC-1029: Routing for Group Signals with Array Access
visbbk	2017-09-19	6.04.00	> STORYC-1971: Support Tx IF ComPdus with multiple PduRSrcPdus
visbbk	2017-09-27	6.05.00	> STORYC-2475: Support Float32 and Float64

visgut	2018-05-25	7.00.00	> STORYC-3089: Respect ComTxModeTimeOffset in combination with Com_SwitchIpduTxMode
visgut	2018-09-10	8.00.00	> STORYC-6351: Support gaps in transformed signal groups (AR 4.3.1) COM
visgut	2019-01-15	8.01.00	> STORYC-1791: Finalize Com TP and Meta Data related features
visgut	2019-03-13	8.02.00	> STORYC-7874: Deny Retry in COM
visgut	2019-03-20	9.00.00	> STORYC-7975: Add doxygen parameter direction
visbbk	2019-10-01	9.00.01	> ESCAN00104132: COM_EXCLUSIVE_AREA_TX is missing for Com_TriggerTransmit
visssg	2020-11-12	10.00.00	> COM-1997: Remove the AUTHOR IDENTIY
viscpe	2020-11-17	10.00.00	> COM-1639: Support repetition cycle without MDT
viscpe	2020-12-03	10.01.00	> COM-1606 Suppress periodic trigger during transmission of mixed mode PDUs with repetitions Step 1
visbbk	2021-01-12	10.02.00	> COM-1848 Introduce Com Units
visbbk	2021-01-15	10.03.00	> ESCAN00108313: Missing Limitation: Com_SendDynSignal during ComIPduCallout
viscpe	2021-01-19	10.03.00	> COM-2282: Remove Com_ReceiveSignal macro API
visssg	2021-02-12	11.00.00	> COM-2215: Support a ComIPdu references to a MainFunction
viscpe	2021-02-25	11.01.00	> COM-2319: Merge Signal Gateway according ASR 4.3 with Multicore trunk
viscpe	2021-02-25	11.01.00	> COM-1954: Update Doc_TechRef with the new template
visssg	2021-03-10	11.02.00	> COM-1849: Implement Com Multipartition - without Signal Gateway
visbbk	2021-04-07	11.03.00	> COM-2523: Remove ComDescriptionRoutingCodeGeneratio n
visbbk	2021-04-27	11.03.00	> COM-2458: Remove Com_IpduGroupControl
visssg	2021-04-29	11.03.00	> COM-2520: Support only I-PDU Callouts with PduInfoPtr

visssg	2021-05-12	11.03.00	> COM-2526: Basic Refactorings for MC Signal Gateway
visbbk	2021-10-18	12.00.00	> COM-1851: Implement Multipartition Com - Description Routing > COM-2885: Implement Multipartition Com - Signal Gateway
alefarth	2022-03-01	13.00.00	> Product name updated to MICROSAR Classic.
vishho	2022-08-16	13.00.00	> COM-3816: Do not support BIG_ENDIAN CPU_BYTE_ORDER == HIGH_BYTE_FIRST > COM-3916: Do not support Gateway
visssg	2023-01-18	13.01.00	> COM-4211: Improve Docu for ArrayAccess
oalka	2023-04-06	14.00.00	> COM-4365: Support FIFO behavior in the event cache
visssg	2023-05-04	14.00.00	> COM-3820: Support BIG_ENDIAN CPU_BYTE_ORDER == HIGH_BYTE_FIRST > ESCAN00114923: Restriction for Immediate Signal Processing is missing for Multipartition Support of the Com.
vadnra	2023-07-07	14.00.00	> COM-4632: Support Gateway > ESCAN111249: Missing Limitation: Rx Invalidation with Replace > COM-3586: Improve Docu for MultiPartition Com and Cross Partition Routing > ESCAN00115239: Missing Limitation: Transfer Property of Group Signals
visms	2024-03-27	14.01.00	> COM-5239: Compile Code as Unity Build

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	Specification of Communication	R4.3.1
[2]	AUTOSAR	Specification of Development Error Tracer	R4.0.3
[3]	AUTOSAR	List of Basic Software Modules	R19-11
[4]	Vector	TechnicalReference MICROSAR Classic RTE	See delivery
[5]	Vector	TechnicalReference MICROSAR Classic Post-Build Loadable	See delivery

This technical reference describes the general use of the Com basis software module.

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

Contents

Illustrations

Tables

1 Introduction

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

Supported Configuration Variants:	PRE-COMPILE [SELECTABLE] POST-BUILD-LOADABLE [SELECTABLE]	
Vendor ID:	COM_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	COM_MODULE_ID	50 decimal (according to ref. [3])

The main purpose of the AUTOSAR BSW module Com is to provide a signal-based interface to the upper layer. In an AUTOSAR based system it is the RTE. In a non-AUTOSAR system it is the application.

It is possible to use the Com layer with different underlying bus systems since they are encapsulated by the PDU Router. Architecture Overview shows how the component is embedded in the AUTOSAR layered architecture.

The main features of the Com component are:

- Provision of interface for signed and unsigned signals to the upper layer
- Packing and unpacking of signals in I-PDUs
- Handling of transmission modes
- Minimum delay between I-PDUs transmissions
- Communication control by starting and stopping of I-PDU groups
- Rx deadline monitoring
- Tx deadline monitoring
- Notification mechanisms
- Initial value support
- Signal Gateway

The implementation is based on the AUTOSAR Com specification [1]. It is assumed that the reader is familiar with this document and other related AUTOSAR specifications.

1.1 Architecture Overview

The next figure shows the interfaces to adjacent modules of the COM. These interfaces are described in chapter 4.

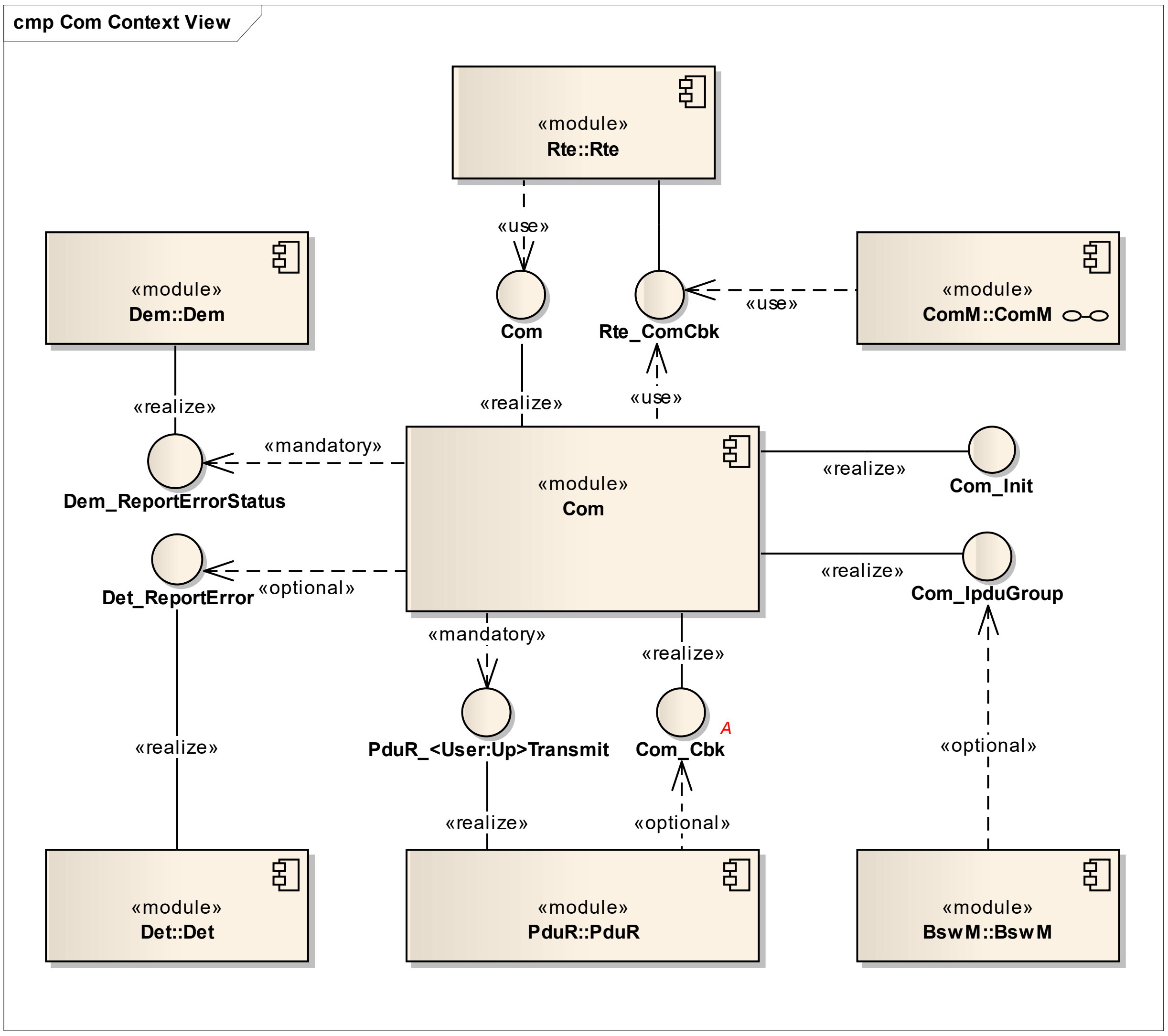

Figure 1-1 Interfaces to adjacent modules of the COM

Applications do not access the services of the BSW modules directly. They use the service ports provided by the BSW modules via the RTE.

2 Functional Description

2.1 Features

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

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

- Table 2-1 Supported AUTOSAR standard conform features
- Table 2-2 Not supported AUTOSAR standard conform features

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
General functionality
Unpacking and packing of Signals and Signal Groups Endianness conversion Sign extension Initialization and De-Initialization services Signal invalidation mechanism Signal status information (update bits) Signal reception filtering Signal Gateway Large data types Dynamic length signals
Communication Modes
Signal Transfer Property I-PDU Transmission Mode Selection of the Transmission Mode for one specific I-PDU Replication of Signal Transmission Requests
Handling of I-PDUs
Starting and stopping of I-PDU groups Minimum Delay Timer
Deadline Monitoring
Reception Deadline Monitoring Transmission Deadline Monitoring
Callouts
I-PDU Callout

Table 2-1 Supported AUTOSAR standard conform features

2.1.1 Deviations

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

Category	Description
Functional	Data sequence control
Functional	Communication protection

Table 2-2 Not supported AUTOSAR standard conform features

2.1.2 Additions/ Extensions

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond The AUTOSAR Standard
Critical section threshold loop strategy
Rx Notification caching
Deferred Event Caching
Handle ID
Strict Repetition Period
Mixed Mode Periodic Suppression

For a detailed description see chapter 2.1.28.

2.1.3 Limitations

2.1.3.1 Component Limitations

Component Limitations
TRUE must be defined to (boolean) 1
FALSE must be defined to (boolean) 0
NULL_PTR must be defined to ((void *)0)

2.1.3.2 Target System

Since 8-bit micro controllers are out of scope in AUTOSAR, COM has been optimized for the usage on 16- and 32-bit controllers. Therefore the target system must be able to provide atomic read and write accesses to 16-bit variables.

2.1.3.3 I-PDU Fan-Out

In case an I-PDU Fan-Out is configured following features are not or partly supported:

- Not supported features
  - Replication of Signal Transmission Requests
  - Transmission Deadline Monitoring
  - Signal Status Information with Update-Bit clear context set to TriggerTransmit

  - Large I-PDUs
  - Confirmation Notification
- Partly supported features
  - Minimum Send Distance of an I-PDU and Minimum Send Distance only for Direct Send Triggers: The I-PDU Fan-Out can lead to an increased Minimum Delay Time as the Minimum Delay Time is reloaded with each confirmation.

2.1.3.4 I-PDU Callout

In case an I-PDU callout is configured and this I-PDU refers to a dynamic length signal, do not call the function Com_SendDynSignal() in the context of the I-PDU callout.

2.1.3.5 Invalidation Mechanism

In case the invalid action of an Rx signal or group signal is configured as REPLACE, normal signal processing including filtering and handling of notifications does not take place.

2.1.3.6 Transfer Property of Group Signals

Every group signal in a Tx signal group must have a configured transfer property. This is a limitation with respect to the AUTOSAR convention that the transfer property of a signal group is considered as the transfer property of its group signals if none of the group signals have a configured transfer property.

2.1.4 Signal Types

The following signal types are supported:

- boolean
- uint8
- uint16
- uint32
- uint64
- sint8
- sint16
- sint32
- sint64
- uint8[n]
- Float32
- Float64

For signed and unsigned integers an endianness conversion is supplied depending on the endianness of the signal and the target system.

The support of signed signals is based on the B-complement.

The data type opaque is interpreted as an unsigned integer and no endianness conversion is performed. The target system specific byte order is applied.

2.1.5 Signal Processing

Each Pdu has the parameter ComIPduSignalProcessing, this parameter can have the value IMMEDIATE or DEFERRED.

> IMMEDIATE signal processing means that notification functions are called within the functions Com_TxConfirmation() or Com_RxIndication().
  - The transmission of a triggered signal with signal processing IMMEDIATE will be triggered within the next call of the respective Com_MainFunctionTx().

> DEFERRED signal processing means that notification functions are called on task level during the next call cycle of the respecitve Com_MainFunctionRx() or Com_MainFunctionTx().
  - Values of signals contained in an I-PDU with signal processing DEFERRED will be updated on task level in the respective Com_MainFunctionRx().

2.1.6 Transmission of a Signal

To request the transmission of a signal, the upper layer uses the API `Com_SendSignal`. After performing optional parameter checks COM updates the I-PDU with the new signal value and checks if the transfer property of the signal requires a direct transmission. If yes, a flag is set which is evaluated later in the cyclic main function of the COM layer's transmit part.

A Tx I-PDU must be assigned to a `Com_MainFunctionTx` .

Transmission modes of the I-PDU are handled in the respective `Com_MainFunctionTx`. This means that the actual transmit request to the underlying layer is always decoupled from the upper layer. In the transmission mode handler cyclic transmissions and direct transmissions are processed.

In the following figure the transmission procedure is shown for I-PDUs with direct and periodic transmission mode.

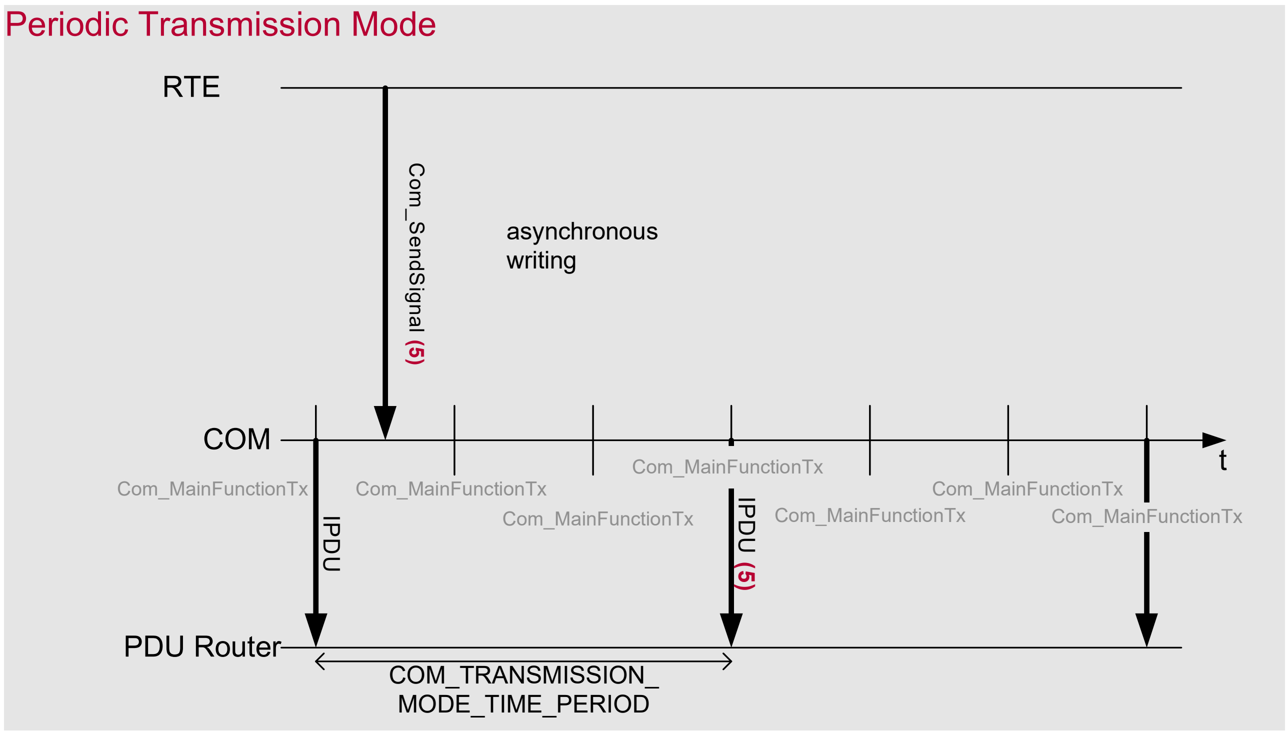

Figure 2-1 Periodic Transmission Mode

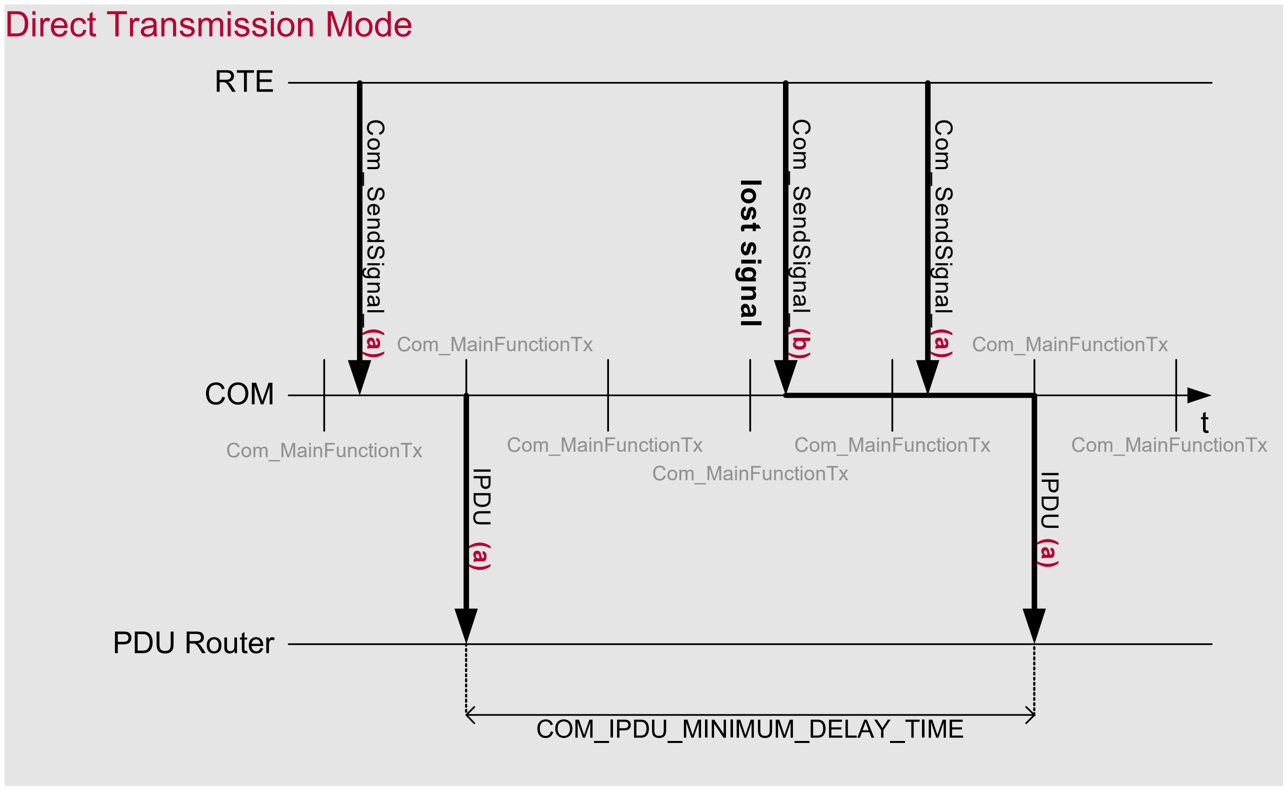

Figure 2-2 Direct Transmission Mode

The mixed transmission mode provides a combination of periodic and direct transmission mode.

Note
Signal values are not queued by COM. If the upper layer updates signals with a higher rate as the I-PDU of the signal is transmitted, only the most recent signal value is sent.

2.1.7 Transmission of a Signal Group

AUTOSAR COM provides signal groups to send several signals consistently. Signals mapped to a signal group are called group signals and should be in relationship with each other. To ensure the consistency of the group signal values, a shadow buffer is provided for each signal group.

To request the transmission of a signal group with several group signals, following sequence of API calls must be followed:

Example
```
/* Update the group signal values in the shadow buffer */
Com_SendSignal(GroupSignal1, &SigBuffer1);
Com_SendSignal(GroupSignal2, &SigBuffer2);
/* Copy the shadow buffer to the Tx buffer */
Com_SendSignalGroup(SignalGroupA);
```

For the transmission modes DIRECT or MIXED the evaluation of the transfer property is handled as follows

- `ComSignalGroup.ComTransferProperty` equals `TRIGGERED` and all `ComGroupSignal.ComTransferProperty` equals `PENDING`
  - `Com_SendSignal(ComGroupSignal) -> Com_SendSignalGroup(ComSignalGroup)` will trigger a transmission of the Tx I-Pdu regardless of the group signal value.
- `ComSignalGroup.ComTransferProperty` equals `TRIGGERED_ON_CHANGE` and all `ComGroupSignal.ComTransferProperty` equals `PENDING`
  - `Com_SendSignal(ComGroupSignal) -> Com_SendSignalGroup(ComSignalGroup)` will trigger a transmission of the Tx I-Pdu if at least one group signal value has changed.
- `ComSignalGroup.ComTransferProperty` equals `PENDING`
  - `ComGroupSignal.ComTransferProperty` equals `TRIGGERED`

  - `Com_SendSignal(ComGroupSignal) -> Com_SendSignalGroup(ComSignalGroup)` will trigger a transmission of the Tx I-Pdu regardless of the group signal value.
- `ComGroupSignal.ComTransferProperty` equals `TRIGGERED_ON_CHANGE`
  - `Com_SendSignal(ComGroupSignal) -> Com_SendSignalGroup(ComSignalGroup)` will trigger a transmission of the Tx I-Pdu if the group signal value has changed.
- `ComGroupSignal.ComTransferProperty` equals `PENDING`
  - `Com_SendSignal(ComGroupSignal) -> Com_SendSignalGroup(ComSignalGroup)` will not trigger a transmission of the Tx I-Pdu.
  -

Caution
To guarantee data consistency of the whole signal group the complete transmission of a signal group (consecutive calls of 'Com_SendSignal' and 'Com_SendSignalGroup') must not be interrupted by another transmission request for the same signal group or by a call of 'Com_InvalidateSignalGroup'.

2.1.8 Transmission Mode Selector

AUTOSAR COM allows configuring two different transmission modes for each I-PDU (ComTxModeTrue and ComTxModeFalse). The transmission mode of an I-PDU that is valid at a specific point in time is selected using only the filter states of the signals that are mapped to this I-PDU.

If a filter of any signal mapped to a specific I-PDU evaluates to TRUE, this I-PDU is transmitted with transmission mode TRUE. The transmission mode FALSE is used for an I-PDU when the filters of all signals mapped to this I-PDU evaluate to FALSE.

If all signals mapped to a specific I-PDU have no filter assigned, the transmission mode evaluates to TRUE and does never change.

The transmission mode is changed as a result of a call of `Com_SendSignal` or `Com_SendSignalGroup`. The value of the signal or group signal that caused the change is already transmitted with the new transmission mode.

In case of an array-based SignalGroup the transmission mode is changed as a result of a call of `Com_SendSignalGroupArray.` The value of the array-based SignalGroup that caused the change is transmitted with the new transmission mode.

By a transmission mode switch to the Direct/N-times transmission mode a direct/ n-times transmission to the underlying layer, respecting the minimum delay time, will be initiated, even if the transmission mode switch was triggered by a signal with PENDING transfer property.

By a transmission mode switch to the Cyclic or Mixed transmission mode the new cycle will start with a transmission request to the underlying layers respecting the minimum delay time.

If the current transmission mode is configured to NONE, the COM will never initiate a transmission to the underlying layer.

2.1.9 Explicit Transmission Mode State Switch

By calling the `Com_SwitchIpduTxMode` API the configured transmission modes can be switched explicitly (TRUE/FALSE) per Tx I-PDU.

If the requested transmission mode is different to the currently active mode, the new transmission mode is immediately active.

- For a new transmission mode PERIODIC or MIXED, the transmission cycle starts with a transmission request, respecting the minimum delay time and the TxModeTimeOffset, and the timer for the cycle time is restarted.
- For a new transmission mode DIRECT or NONE, no transmission of the Tx I-PDU is triggered by the API call.

If the requested TMS is already active, the function call will silently be ignored.

By mixing the signal based TMS switch and explicit TMS switch by `Com_SwitchIpduTxMode` for the same I-PDU, the signal based TMS switches the manual set mode back, during a call to `Com_SendSignal` or `Com_SendSignalGroup` for this Tx I-PDU.

2.1.10 Transmit Signal Filters

A signal filter can be optionally assigned to each transmit signal. The filter of a transmit signal is only used for transmission mode selection but the value of a transmit signal is never filtered out.

The following filters are supported:

F_Always	(TRUE)
F_Never	(FALSE)
F_MaskedNewDiffersMaskedOld	((*new_value*&*mask*) != (*old_value*&*mask*))
F_MaskedNewEqualsX	((*new_value*&*mask*) == *x*)
F_MaskedNewDiffersX	((*new_value*&*mask*) != *x*)
F_MaskedNewIsOutside	((*new_value*<*min*) || (*max*<*new_value*))
F_MaskedNewIsWithin	((*min*<=*new_value*) && (*new_value*<=*max*))

The values for *mask*, *x*, *min* and *max* can be configured for each filter.

2.1.11 Minimum Send Distance of an I-PDU

In the COM specification an optional mechanism is defined to achieve a delimitation of the bus load, by introducing a minimum send distance for an I-PDU. This concept is also handled in the Tx main function.

In the following figure an example for the mixed transmission mode is shown. Note that due to the minimum send distance, cyclic transmissions can be delayed, however the base cycle is not modified. Direct transmissions are drawn by solid red arrows.

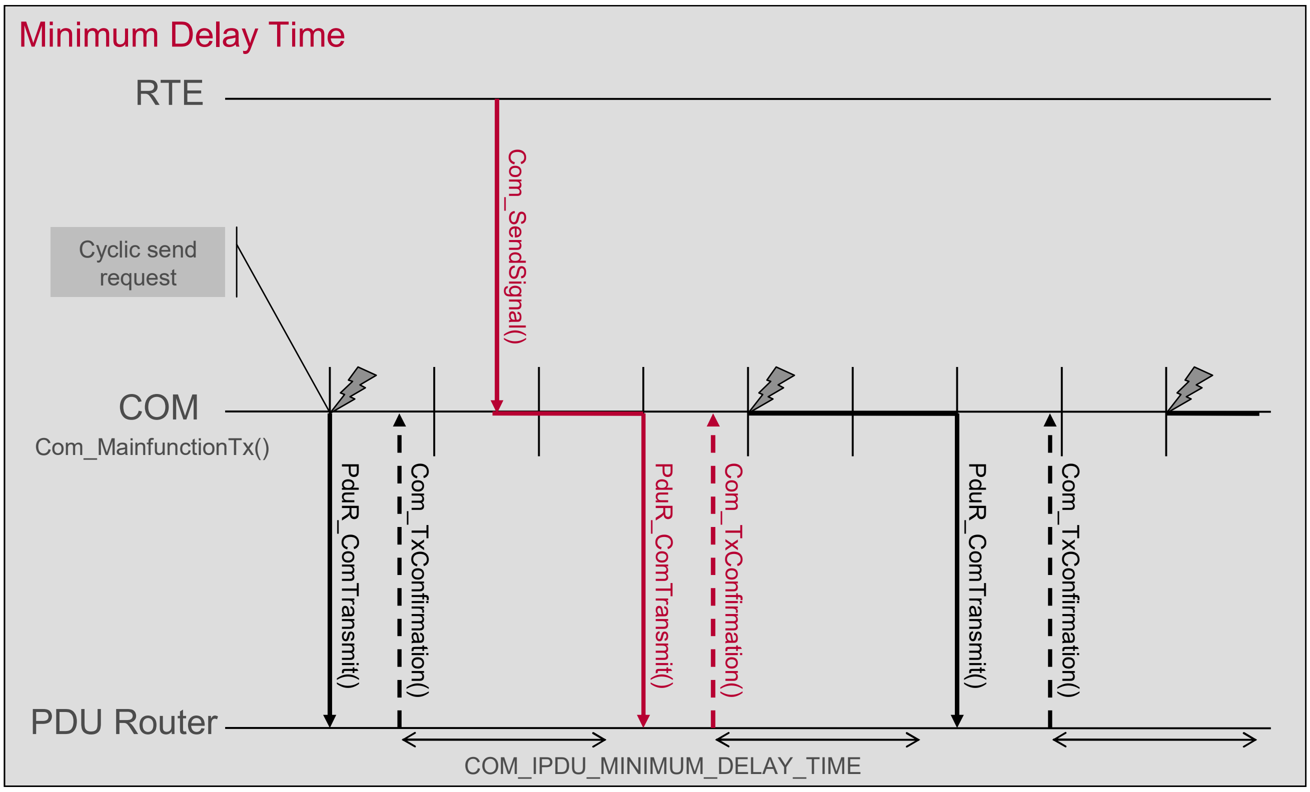

Figure 2-3 Minimum Delay Time

The handling of the minimum send distance is based on the evaluation of the confirmation by the underlying layer.

2.1.12 Minimum Send Distance only for Direct Send Triggers

If the parameter ComEnableMDTForCyclicTransmission is set to false, the Minimum Delay Time will only be considered for event based transmissions, which can be initiated by `Com_InvalidateSignal()`, `Com_InvalidateSignalGroup()`, `Com_SendSignal()`, `Com_SendSignalGroup()`, `Com_SendSignalGroupArray()` or `Com_TriggerIPDUSend()`.

Figure 2-4 shows that cyclic transmissions do not wind up the Minimum Delay Time. An event based transmission is directly triggered after a cyclic transmission although the minimum delay has not elapsed. Right after, in comparison a second event based transmission request is delayed by the configured Minimum Delay Time as the first event based transmission has reloaded the delay counter.

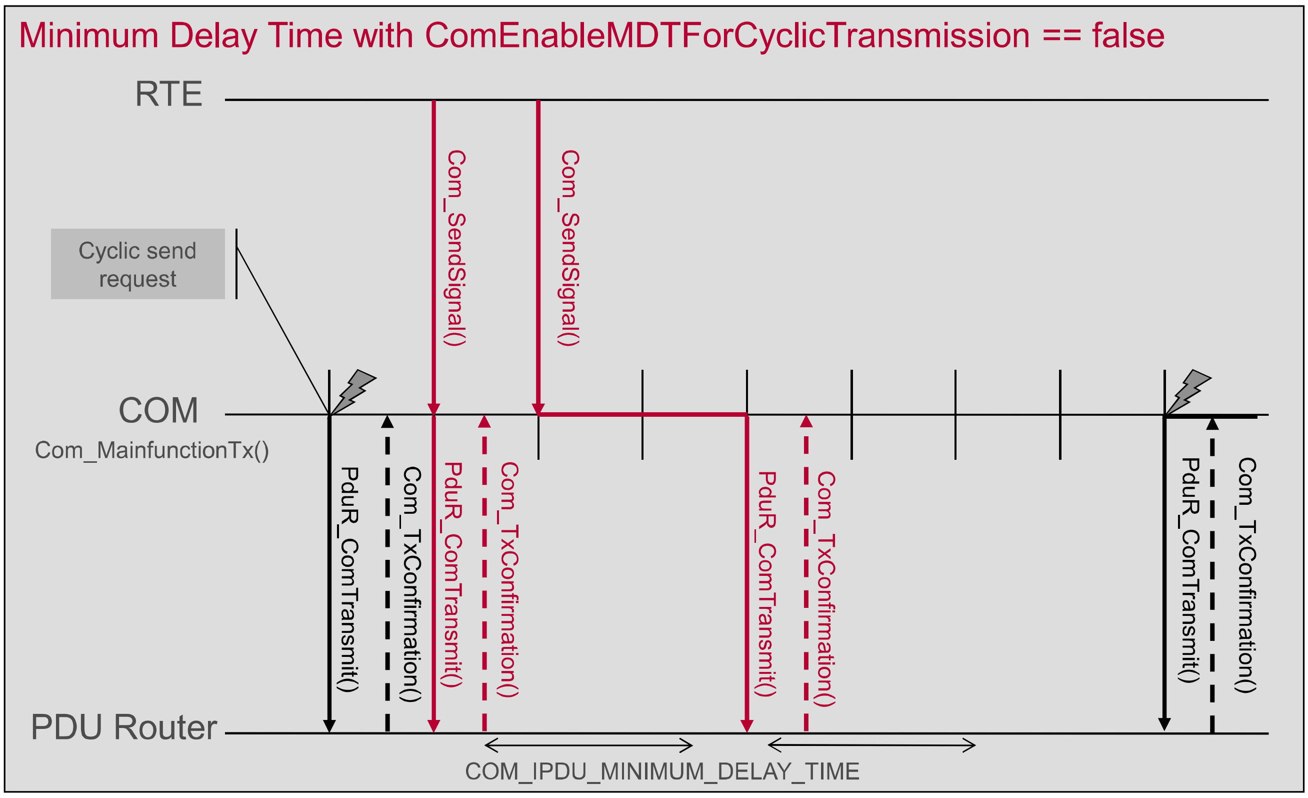

Figure 2-4 Minimum Delay Time with ComEnableMDTForCyclicTransmission == false

2.1.13 Transmission Deadline Monitoring

For Tx I-PDUs a deadline monitoring mechanism is provided to detect failures in the transmission mechanism of the lower layers.

Two different variants are supported:

- Normal Mode: If the COM triggers the transmission of the I-PDU, the time between the send request for the I-PDU and the next Tx confirmation (`Com_TxConfirmation()`) is observed. A send request for example is given by Com_SendSignal(), Com_TriggerIPduSend() or cyclic triggers.
- None Mode: For I-PDUs triggered by a schedule table of a bus interface (e.g. LIN schedule table), the time between two consecutive Tx confirmations (`Com_TxConfirmation()`) is observed. The “None Mode” is applied for Tx I-PDU with both transmission modes configured to NONE. The timer is started whenever the corresponding I-PDU Group of the I-PDU is started and reloaded on a transmission confirmation. However, a trigger event for example given by Com_TriggerIPduSend() also start’s the timer, if it’s not already running.

The transmission deadline monitoring for the “Normal Mode”, is illustrated in Figure 2-5. Each time a signal or signal group is written, the timeout timer is started if it is not yet running. If the timeout time (configuration parameter `ComTimeout`) is expired before the next Tx confirmation is received, all configured timeout notification functions of the I-PDU are called.

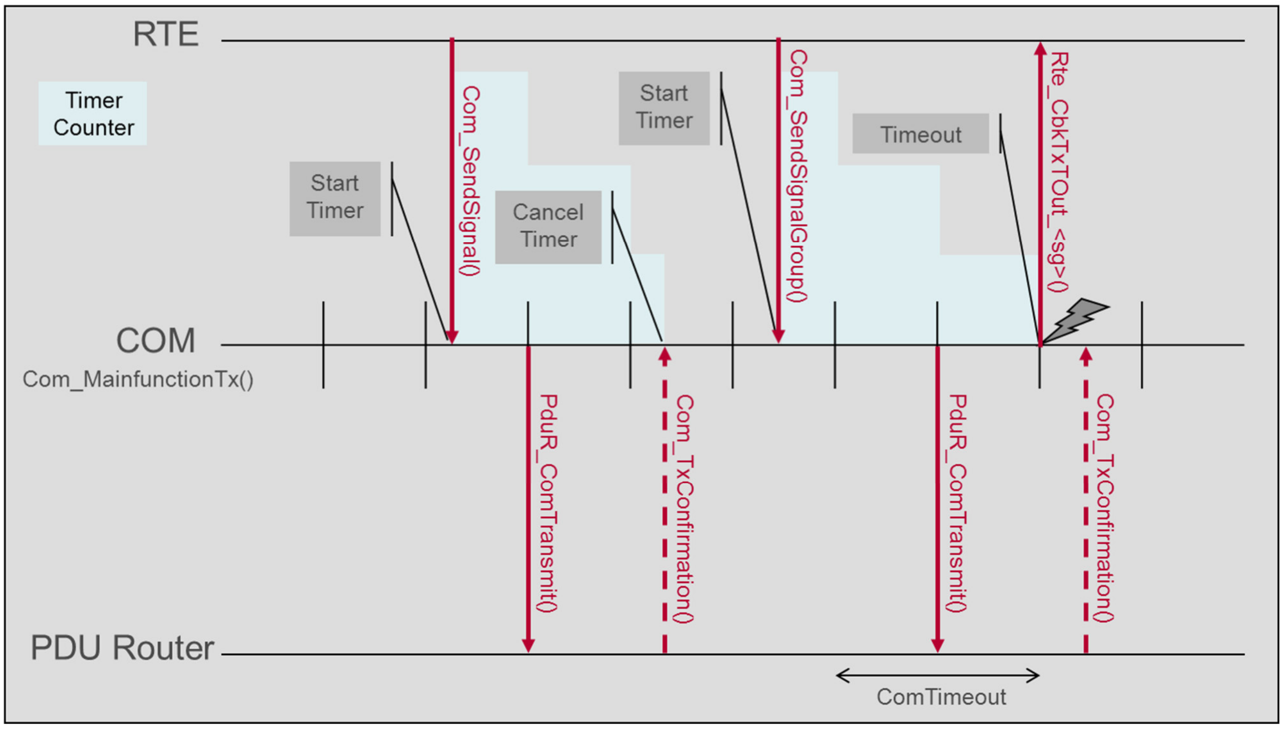

Figure 2-5 Transmission Deadline Monitoring – Normal Mode

In the “None Mode” the timeout timers are initially started by the start of the corresponding I-PDU group and are restarted each time a Tx confirmation is received. Figure 2-6 illustrates this behavior.

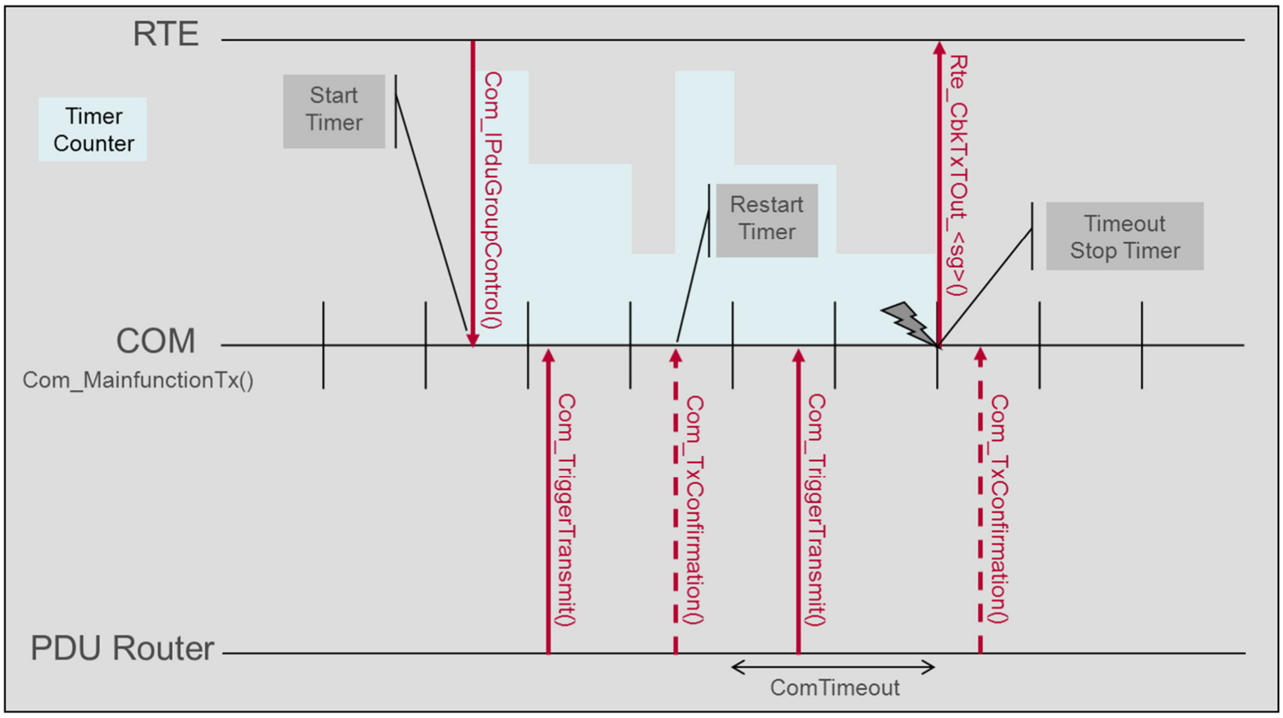

Figure 2-6 Transmission Deadline Monitoring – None Mode

2.1.14 Replication of Signal Transmission Requests

The AUTOSAR COM provides an optional feature to replicate transmission requests to the lower layer for one send request by an upper layer.

If the attribute ComTxModeNumberOfRepetitions is configured to a value ‘n’ greater than ‘0’ for a transmission mode DIRECT or MIXED, the COM triggers the transmission of the Tx I-PDU cyclically with a configurable ComTxModeRepetitionPeriodFactor as long as ‘n + 1’ confirmations for this I-PDU are invoked after a send request by an upper layer.

Figure 2-7 illustrates this behavior exemplarily for ‘n = 2’ replications with a repetition period factor ‘t_d’ configured to ‘2’.

Note
Under certain conditions, if the confirmation is invoked late, the number of messages send out on the bus can differ from the configured number of repetition, as the confirmations and not the transmission requests are counted.

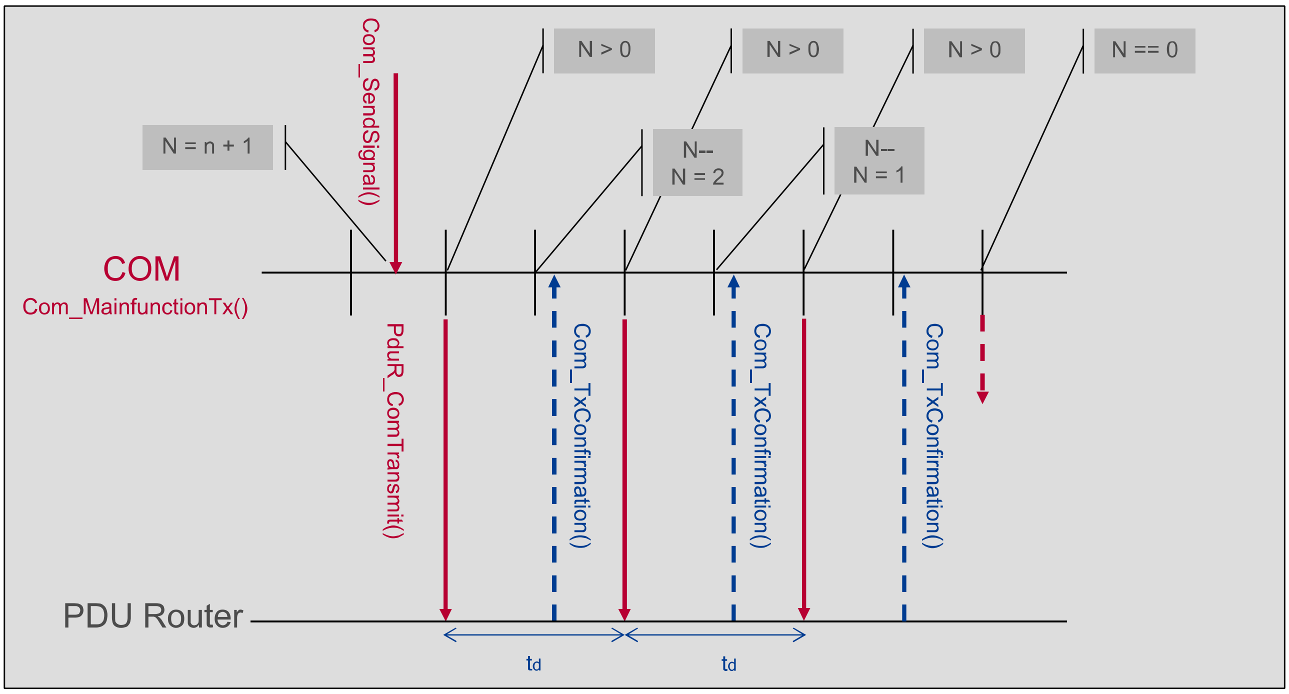

Figure 2-7 Replication of Signal Transmission Requests for n = 2 repetitions

As the replications are an attribute of a Tx I-PDU, each send request of a mapped signal or signal group with a transfer property ‘TRIGGERED’ or ‘TRIGGERED_ON_CHANGE’ provokes the replications of the Tx I-PDU.

To enable replication control per signal following transfer properties were introduced

- ‘TRIGGERED_WITHOUT_REPETITION’
- ‘TRIGGERED_ON_CHANGE_WITHOUT_REPETITION’

- A send request for signals with these transfer properties does only initiate one single transmission requests to the lower layer even if replications are configured for the current transmission mode.

Note

- If a *-WITHOUT-REPETITION signal and a normal triggered signal are written at the same time, the repetitions are triggered.
- If a *-WITHOUT-REPETITION signal is written while repetitions of a former send request are processed, the outstanding transmission repetitions are not canceled and an additional transmission is triggered.

2.1.15 Reception of a Signal

To receive a signal the upper layer uses the API `Com_ReceiveSignal`. This service delivers the signal value which is contained in the latest I-PDU of the signal.

As the signal processing context depends on the configuration of the corresponding Rx I-PDU, the latest signal value might not be available until the next call to the respective `Com_MainfunctionRx`.

An Rx I-PDU can be assigned to a `Com_MainfunctionRx` depending on the signal processing context and the reception deadline monitoring. Only for deferred processing Rx IPDUs and for I-PDUs with configured reception deadline moitoring a mapping of an Rx-IPDU to a `Com_MainfunctionRx` is necessary.

The reception procedure of the signal is usually asynchronous to the reception of the I-PDU. It is however possible to call `Com_ReceiveSignal` in the reception notification callback.

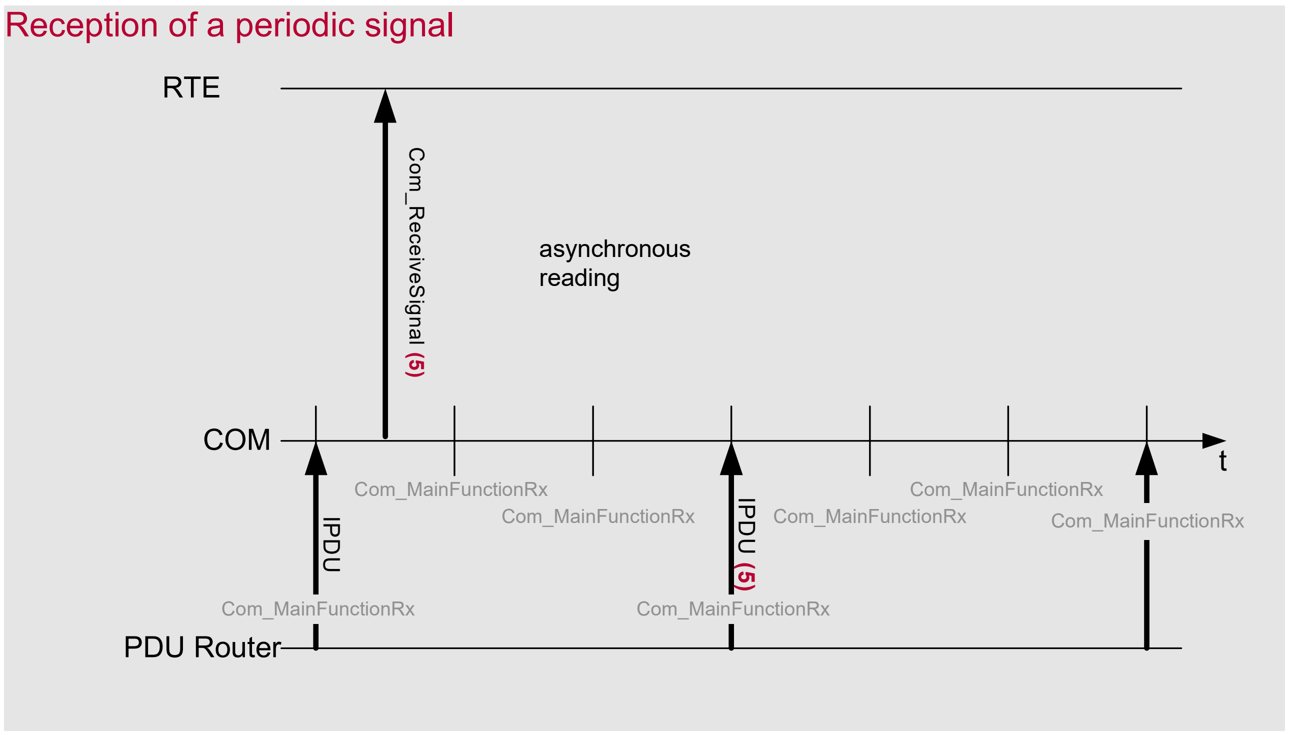

Figure 2-8 Reception of a periodic signal

A call to `Com_ReceiveSignal` always returns the last received signal value or the initial value if a timeout occurred and the Rx Data Timeout Action is set to REPLACE, even if the corresponding I-PDU group is stopped.

2.1.16 Reception of a Signal Group

AUTOSAR COM provides signal groups to receive several signals consistently. Signals mapped to a signal group are called group signals and should be in relationship with each other. To ensure the consistency of the group signal values a shadow buffer is provided for each signal group.

As the signal processing context depends on the configuration of the corresponding Rx I-PDU, the latest signal group value might not be available until the next call to the respective Com_MainfunctionRx.

To receive the values of a signal group with several group signals, following sequence of API calls must be followed:

Example

```
/* Copy the Rx buffer to the shadow buffer */
Com_ReceiveSignalGroup(SignalGroupA);
/* Get the group signal values from the shadow buffer */
Com_ReceiveSignal(GroupSignal1, &SigBuffer1);
Com_ReceiveSignal(GroupSignal2, &SigBuffer2);
```

Caution
To guarantee data consistency of the whole signal group the complete reception of a signal group (consecutive calls of 'Com_ReceiveSignalGroup' and 'Com_ReceiveSignal') must not be interrupted by another reception request for the same signal group.

2.1.17 Array-based access of SignalGroups

An array-based access of SignalGroups represents an alternative to the aforementioned approaches in 2.1.7 and 2.1.16 to access SignalGroups. Instead of treating all GroupSignals individually, SignalGroups are accessed as serialized composite data. The APIs Com_ReceiveSignalGroupArray and Com_SendSignalGroupArray are used to send and receive the uint8-array based representation of the SignalGroup. Using this feature, the overhead for packing and unpacking the GroupSignals individually vanishes and further processing of the SignalGroup is left to the caller of the API. However, to permit fast processing, following preconditions have to be fulfilled:

- Only fix-sized data types are supported.
- The SignalGroup must be byte-aligned within the containing I-PDU.
- The SignalGroup must not be intermitted by other signals. However, gaps may be present within the SignalGroup.
- Only ALWAYS, NEVER, MASKED_NEW_EQUALS_X and MASKED_NEW_DIFFERS_X filters are supported for TMS.
- Data Invalidation and the Timeout Action Replace are not supported.

To activate this feature, the global COM configuration switch ComEnableSignalGroupArrayApi must be enabled. Further, only SignalGroups with ComSignalGroupArrayAccess being activated are permitted for the array-based access.

Note
If a group signal belonging to a signal group with enabled array access is routed via signal gateway, the above mentioned advantages of the array based representation will vanish as the signal gateway requires unpacking the group signals.

2.1.18 Dynamic DLC

The COM evaluates the actual received DLC of the SDU given from the lower layer interface to support the reception of Rx I-PDUs with a variable length.

Two cases are distinguished:

- Actual received DLC is greater than or equal to the statically configured
  - Only the SDU payload data with the statically configured PDU length is processed.
  - Normal signal processing.
- Actual received DLC is smaller than the statically configured
  - Only the SDU payload data with the actual received PDU length is processed.
  - Only completely received signals or signal groups are processed.
    This affects:
    - Rx indication notifications
    - Rx Filter
    - Rx Invalidation
    - Signal routing
  - If a configured update-bit is not contained in the actual received payload, the signal is processed as if the update-bit were set.

Note
If a signal or signal group is completely received depends on the following parameters:

> Bit position and length

> Endianness

> The position of all group signals of a signal group

2.1.19 Reception Deadline Monitoring

For Rx signals and signal groups, a deadline monitoring mechanism is provided to detect failures of other ECUs.

The timeout time can be configured per signal and signal group, but only signals and signal groups with a configured update-bit can be monitored separately. For signals and signal group without a configured update-bit, an I-PDU based timeout is applied.

If a timeout occurs, it’s configurable whether the COM shall call a timeout notification and additionally replaces the signal value with the configured initial value. If the Rx Timeout value should differ the initial value, the parameter “Rx Data Timeout Substitution Value” can be used to configure an Rx timeout substitution value.

The start of the timeout monitoring can be deferred by using the first timeout time to avoid timeout events in the startup time of a network.

If no first timeout time is configured, the deadline monitoring is not started until the first reception of the Rx I-PDU or a set update-bit. After the first reception, the normal timeout time is monitored.

2.1.20 Invalidation Mechanism

On sender side an invalid value can be configured per signal and group signal which is set by the COM invalidation APIs to indicate than no valid signal value can be provided by the application.

For signal groups all group signals should be invalided at once, because the group signals are related in a consistent manner. Thus, if one group signal is invalid, the whole signal group is invalid.

On receiver side the COM checks the value of received signals and group signals against the configured invalid value. If an invalid value is detected, the COM offers the following actions:

- The invalid value is replaced by the initial value of the signal. Normal signal processing including filtering and handling of notifications does not take place.
- A configured invalid notification is called and the invalid value is not stored in the internal COM buffer.
- For signal groups, all group signals are checked against their invalid value. If at least one group signal is invalid, the invalid action is performed for all group signals of the signal group.

2.1.21 Signal Reception Filtering

A filter algorithm can be configured for Rx signals and group signals to filter out specific signal values. If the filter algorithm is evaluated to FALSE, the complete signal processing is inhibited. Thus the signal data is not stored in the internal COM buffer and a configured notification function is not called.

For signal groups the signal processing is performed, if at least one configured filter algorithm of any of the contained group signals is evaluated to TRUE. The signal group is only filtered out, if all filter algorithms are evaluated to FALSE.

2.1.22 Signal Status Information

For signals and signal groups update-bits can be configured to indicate whether the signal value has been updated by the application since the last transmission of the signal on the bus.

On sender side the update-bit is set in the context of Com_SendSignal or Com_SendSignalGroup. As the update-bit shall only be present on the BUS once after the value is updated, the update-bit has to be cleared dependent on the send behavior of the lower layer. As the COM shall be aware of the lower layer, the clear context of the update-bit is configurable per Tx I-PDU.

If the lower layer copies the I-PDU payload in the transmit context, the update-bits shall be cleared directly after the COM triggers the transmission of the I-PDU. In case the lower layer requests the I-PDU payload decoupled by a call to Com_TriggerTransmit, the update-bits shall be cleared in this context.

The receiving ECU checks the state of an update-bit associated to a signal or signal group and inhibits the complete signal processing, if the update-bit is not set. In a well-functional network, the update-bit should always be set if the signal value has changed. Otherwise the sending ECU is defect.

2.1.23 Signal Gateway

The signal gateway allows routing of signals and signal groups from an Rx I-PDU to one or several Tx I-PDU(s). To reduce interrupt runtime, signal routing is executed on task level within Com_MainFunctionRouteSignals().

As signals can be accessed individually by COM, it is possible to change the signal layout of I-PDUs while routing. Furthermore it is possible to change the byte alignment of the routed signals and to specify any available transmission property for the Tx signal and I-PDU.

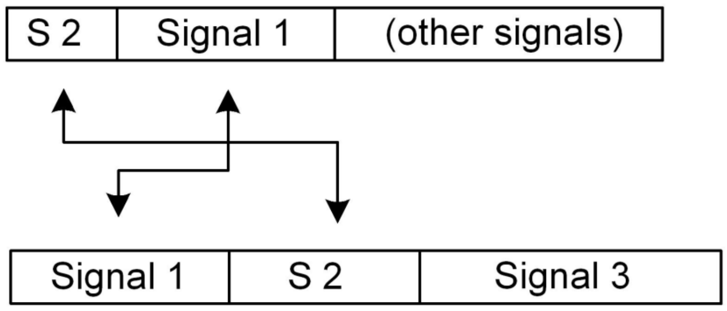

Figure 2-9 Signal routing allows routing of individual signals

2.1.23.1 Signal routing requirements

In order to allow routing between two signals several requirements must be fulfilled regarding the compatibility of the Rx and the Tx signal:

- Application data type must be equal
- Rx signal bit count must not be larger as the Tx signal bit count
- When routing byte arrays (application data type is uint[8]) the byte count must be equal
- If a Tx signal is used in multiple routing relations (n(Rx):1(Tx) routing) the routing relations must be exclusive at runtime to ensure data consistency.

2.1.23.2 Routing of signal groups

Signal groups are routed consistently from the Rx I-PDU to the Tx I-PDU. In order to allow data consistency of the signals, it is required that all signals of the Tx signal group are filled with signals of one Rx signal group. If this is not given, the signal group routing is not possible.

It is not required that all signals of an Rx signal group are routed to a Tx signal group. This allows routing an Rx signal group to a Tx signal group with less group signals.

2.1.23.3 Routing latency for normal Signal Gateway

The maximum routing latency is influenced by several factors and cannot be guaranteed by COM. When estimating the routing latency, the following factors have to be taken into account:

- the cycle times of COM (main) functions
- the minimum delay time of the Tx I-PDU
- the Tx I-PDU transmission mode and cycle time
- the Tx signal transfer mode and filter settings
- delays caused by lower layers (such as bus access delay times)
- other factors such as interrupt events that can delay routing execution

Example

Eliminating all these factors to the fastest possible configuration (minimum delay time is zero, Tx I-PDU send mode is DIRECT, Tx signal transfer property is ‘TRIGGERED’, signal processing is immediate …) the maximum routing latency is the sum of the following cycle times:

Call cycle of Com_MainFunctionRx(): If the signal processing of the Rx I-PDU is configured to deferred, the routing event flag is evaluated deferred by this main-function. For immediate signal processing the flag is evaluated in context of Com_RxIndication().

Call cycle of Com_MainFunctionRouteSignals(): This task polls the routing event flag. If the flag is set the signal is copied to the destination I-PDU and the transmission request flag of the Tx I-PDU is set.

Call cycle of Com_MainFunctionTx(): This function is used to control the transmission of I-PDUs. The function polls the transmission request flags and triggers the I-PDU transmission by calling PduR_ComTransmit() of the related Tx I-PDU.

2.1.23.4 Gateway routing timeout

The Tx-I-PDU based gateway routing timeout describes the maximum time between two routing events that refer the same Tx-I-PDU, before a timeout occurs. If a routing timeout occurs, the cyclic transmission of a Tx-I-PDU is stopped. The cyclic transmission of the PDU is reinitiated if any gateway mapping event occurs for the Tx-I-PDU.

2.1.23.5 Signal Processing

From Autosar version 4.3 and onwards routed signals shall be exempt from:

- Timeout action
- Reception filtering
- Invalid action

A separate Buffer is used to achieve this behaviour (see (1) in Figure 2-10 Gateway signal processing).

The original behaviour, where a routed signal undergoes the same processing as a signal received by the upper layer, can still be recreated (see (2)).

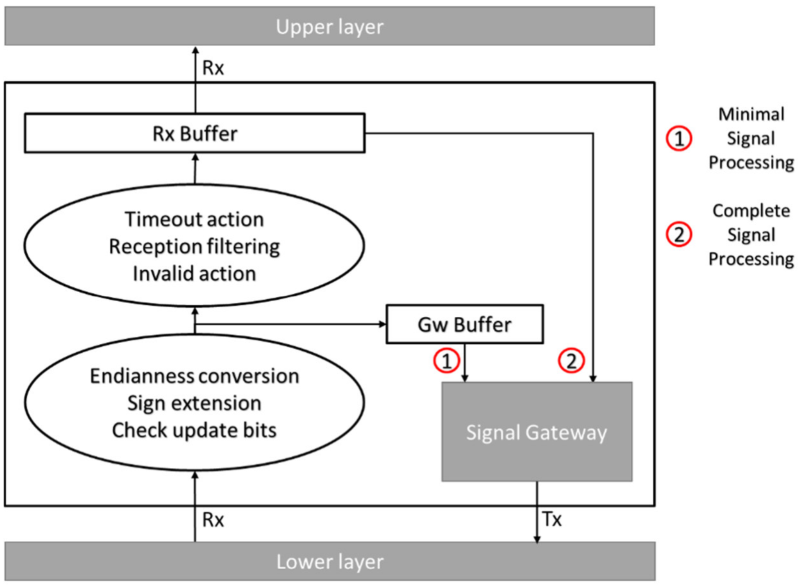

Figure 2-10 Gateway signal processing

Which of these modes is used is defined by the ComSignalGateway general switch.

2.1.24 Gateway Description Routing

Description routing represents a basic routing mode, in which a portion of an incoming Rx-I-PDU is copied to 1…n destination Tx-I-PDUs without any further interpretation and processing of the content. In this mode, the normal signal processing path is bypassed allowing a reduction of the routing event latency.

A minimal set up of a gateway mapping consists of one source and one destination description. A source description is at least defined by a source I-PDU reference, start bit position, bit size and endianness. The bit size defines the amount of bits that should be copied starting from the start bit position. On the other hand, a destination description is defined by a destination I-PDU, a start bit position and endianness. Additionally, a transfer property can be configured.

The usage of this gateway mode requires several preconditions to be fulfilled:

- Source and destination description of a gateway mapping share the same endianness.
- Sign extension, endianness conversion and filtering are not supported.

The call context and the point of time, when a gateway mapping is processed after reception, are defined through the source and destination I-PDU signal processing property:

		Destination I-PDU	
		Immediate	Deferred
Source I-PDU	Immediate	Rx and Tx: Com_RxIndication()	Rx: Com_RxIndication() Tx: Com_MainFunctionTx()
	Deferred	Rx and Tx: Com_MainFunctionRouteSignals()	Rx: Com_MainFunctionRouteSignals() Tx: Com_MainFunctionTx()

Table 2-3 Gateway mapping processing context

A gateway mapping in which source and destination descriptions refer I-PDUs with an immediate signal processing property, have the least routing latency. In this case, copying the referred bits from Rx- to Tx-I-PDU buffer and transmit initiation take place in the context of Com_RxIndication().

Note
If multiple source I-PDUs have a deferred signal processing property, then the received I-PDUs are processed in reverse order.

2.1.25 Cross Partition Gateway Routing

The Signal Gateway and Description Routing features allow the routing of signals or descriptions from source Com I-PDUs associated with a given partition to one or more destination Com I-PDUs associated with a different partition.

When cross partition gateway routings are processed, received data is copied into a CrossPartitionRoutingQueue. For each routing direction, a queue is used to exchange data. The source partition has write access to the queue and the destination partition has read access to the queue. The queue is read from the destination partition in the context of the Com_MainFunctionRouteSignals().

The CrossPartitionRoutingQueue consists of two unidirectional cross partition queues. It is possible to specify the queue size for each direction independently. The data written into the queue consists of a header byte, an index value specifying a particular gateway routing, and the data of the received signal/description. The size of the index value ranges from 1 to 4 bytes based on the least representation to address the configured routings.

Note
Carefully choose values for CrossPartitionRoutingQueue sizes if this feature is used. The queue size must be chosen to buffer all received signals/descriptions associated with cross partition routings in respective partitions. This size must also include the overhead bytes described above associated with each routing. In case of overflow, the queue retains already buffered values and does not allow further entries to be added.

2.1.26 Large I-PDUs

A large I-PDU is a PDU that is too large to fit into a single L-PDU of the underlying communication protocol.

A large I-PDU must have the ComIPduType configured to TP and will be transmitted/received by a transport protocol.

On Receiver side the COM holds for always a valid value of the signals contained in a large I-PDU.

On Transmission side the COM will return COM_BUSY, by a call of the APIs: Com_SendSignal, Com_SendSignalGroup and Com_SendDynSignal when a large I-PDU transmission is in progress.

Note
For large I-PDUs the ComTxIPduClearUpdateBit context can only be configured to Confirmation.

2.1.27 Dynamic length signals

Dynamic length signals are ComSignals or ComGroupSignals with a ComSignalType configured to UINT8_DYN. The range of the length of a dynamic length signal is 0 to the configured ComSignalLength.

Dynamic length signals must be contained in an I-PDU with the ComIPduType configured to TP and dynamic length signals must be placed at the end of the I-PDU.

It is allowed to configure an update-bit for a dynamic length signal. In this case, the update-bit must be located in front of the dynamic length signal.

Use the API Com_ReceiveDynSignal to receive a dynamic length signal and use the API Com_SendDynSignal to send a dynamic length signal.

Dynamic length signal must be placed to byte boundaries and must have the signal endianness OPAQUE.

On reception side the only supported ComFilterAlgorithm is ALWAYS, whereas on transmission side the ComFilterAlgorithms ALWAYS and NEVER are supported for dynamic length signals and group signals. Further, the ComTransferProperties TRIGGERED_ON_CHANGE and TRIGGERED_ON_CHANGE_WITHOUT_REPETITION are not supported for dynamic length signals and group signals.

2.1.28 Com Optimizations

2.1.28.1 Critical section threshold loop strategy

Critical sections as described in chapter 3.2 are used to avoid concurrency/ data consistency problems when shared ressources are used. However, switching the state of critical sections might be an expensive operation. To optimize the relation between runtime with locked interrupts and the cost introduced by entering and exiting an exclusive area a threshold strategy is applied when multiple elements are processed in a loop.

This strategy locks the desired exclusive area before entering loop and exits it after all elements have been processed. Further, at the end of each iteration step a counter is incremented and compared with configured threshold value. If the counter exceeds the threshold the exclusive area will be temporarily be opened to allow rescheduling of waiting tasks as shown in following example.

Example

```
Com_EnterExclusiveArea();
for(; idx < tableSize; idx++)
{
  exclusiveAreaCounter++;
  /* Do processing */
  if(exclusiveAreaCounter >= exclusiveAreaThreshold)
  {
    exclusiveAreaCounter = 0;
    Com_ExitExclusiveArea();
    Com_EnterExclusiveArea();
  }
}
Com_ExitExclusiveArea();
```

Following thresholds can be configured:

Threshold (Affected Exclusive Area)	**Description**
ComGatewayDescriptionProcessingISRLockThreshold (COM_EXCLUSIVE_AREA_BOTH)	Strategy is applied on gateway description processing, where the source description is deferred.
ComGatewayProcessingISRLockThreshold (COM_EXCLUSIVE_AREA_BOTH)	Strategy is applied in context of Com_MainFunctionRouteSignals, where gateway signal routing elements are being processed.
ComIPduGroupISRLockThreshold (COM_EXCLUSIVE_AREA_RX/ COM_EXCLUSIVE_AREA_TX)	Strategy is applied in context of Com_IpduGroupStart / -Stop respectively. Threshold describes the max. number of Rx or Tx I-PDU's which are being started or stopped before ISR Locks will temporarily be opened.
ComRxProcessingISRLockThreshold (COM_EXCLUSIVE_AREA_RX)	Threshold describes the max. number of Rx I-PDUs which are being processed after reception in context of the respective Com_MainFunctionRx. Note: Threshold might be released temporarily if a notification/ or invalid notification has to be called, whenever the deferred notification cache is full (see section 0).
ComTxProcessingISRLockThreshold (COM_EXCLUSIVE_AREA_TX)	Threshold describes the max. number of Tx I-PDUs which are processed for transmission under locked interrupts in context of the respective Com_MainFunctionTx.

Table 2-4 Configurable Thresholds

2.1.28.2 Rx Notification caching

A RTE/ application callback must always be called with open interrupt locks, therefore it might be required to temporarily exit an previously entered exclusive area. To reduce the cost of switching the state of an exclusive area, notification callbacks and invalid notification callbacks are cached in a configurable notification cache while signals and signal groups are being unpacked after reception. The idea behind this strategy is, instead of exiting and reentering the Rx exclusive area multiple times, to call all cached notifications afterwards at once, after all received signals/ signal groups have been processed. However caching requires some amount of memory, therefore the user can configure the cache size. Whenever, the cache is full, the exclusive area will be temporarily exited and all cached notifications and invalid notifications callbacks will be called.

Depending on the configured signal processing property of the I-PDU, the callbacks will be either be called in immediate or deferred notification cache:

- Immediate notification cache: Callbacks will be cached in the context of Com_RxIndication. Cache is reserved on the stack whenever an immediate I-PDU is being received. Therefore the cache has a local scope for each received immediate I-PDU.
- Deferred notification cache: Callbacks will be cached in the task context of the respective Com_MainFunctionRx. One shared cache is reserved on the heap and therefore has a global scope for all deferred I-PDUs. Signal/ signal group callbacks of all deferred I-PDUs will be cached before the callbacks will be called.

Note
User has to ensure that the configured stack size complies with the configured immediate notification cache and reception rate of immediate I-PDUs.

2.1.28.3 Deferred Event Caching

This feature describes one optimization strategy for reducing the processing time of deferred I-PDUs. The main idea behind this feature is to cache the ID of received Rx-I-PDUs that should be processed in deferred manner and therefore avoid the handling of all configured deferred I-PDUs. The optional parameter ComRxDeferredEventCacheSize describes a cache size for storing the amount of deferred I-PDUs. The maximum size of the cache is limited to the number of configured I-PDUs. If the number of deferred events exceeds the cache size, the deferred I-PDUs will be processed in normal fashion and every configured I-PDU has to be checked if a new event has occurred. This feature can be activated by enabling setting the paraemter ComRxDeferredEventCacheSize to a value bigger than zero.

Note
If the deferred event caching strategy is used first-in-first-out behavior applies.

2.1.28.4 Handle ID

All data unit elements (Signal, SignalGroup, GroupSignal, I-PDU, I-PDU Group) contain a numerical ID which is unique for the type of the data unit. This Handle ID is required to access these elements via respective API calls.

However, Signals, SignalGroups and GroupSignals can be defined without being accessed from outside. In this case, the assignment of a Handle ID is obsolete and can be removed to reduce the computational overhead and code size. Though these unit elements cannot be accessed, they are considered for the calculation of the initial values of their containing I-PDU. The allocation of the Handle ID is coupled to the values of parameters ComSignalAccess and ComSignalGroupAccess. These parameters determine whether the Handle ID is required or not:

ComSignalAccess/ ComSignalGroupAccess	Description	Handle ID
ACCESS_NEEDED_BY_SWC_OR_COM	A data or a gateway mapping is present.	x
ACCESS_NEEDED_BY_OTHER	A Handle ID is required by user or any other module.	x
ACCESS_UNCLEAR	It is unclear, if the data unit element needs to be accessed.	x
ACCESS_NOT_NEEDED	Handle ID is not needed, as neither a data nor a gateway mapping could be found.	

It should be noted, if the Handle ID of a GroupSignal is required, the containing SignalGroup must have a Handle ID as well. Further, for an array-based access of SignalGroups, a Handle ID for the SignalGroup and all GroupSignals is essential.

2.1.28.5 Strict Repetition Period

If the MDT is still active because of a previous transmission a transmission will be delayed to a later main function. This delay does not apply for the timer that tracks repetitions because it is always started in the main function following a direct trigger. This can lead to the actual distance between two PduR_ComTransmit being smaller than the configured repetition period. This behavior (depicted below in blue) is Autosar 4.1.3 conform.

If the behavior from Autosar 4.0.3 and earlier is desired the general switch ComStrictRepetitionPeriod can be enabled. The timer for repetitions is then started when the actual PduR_ComTransmit is done (depicted below in red).

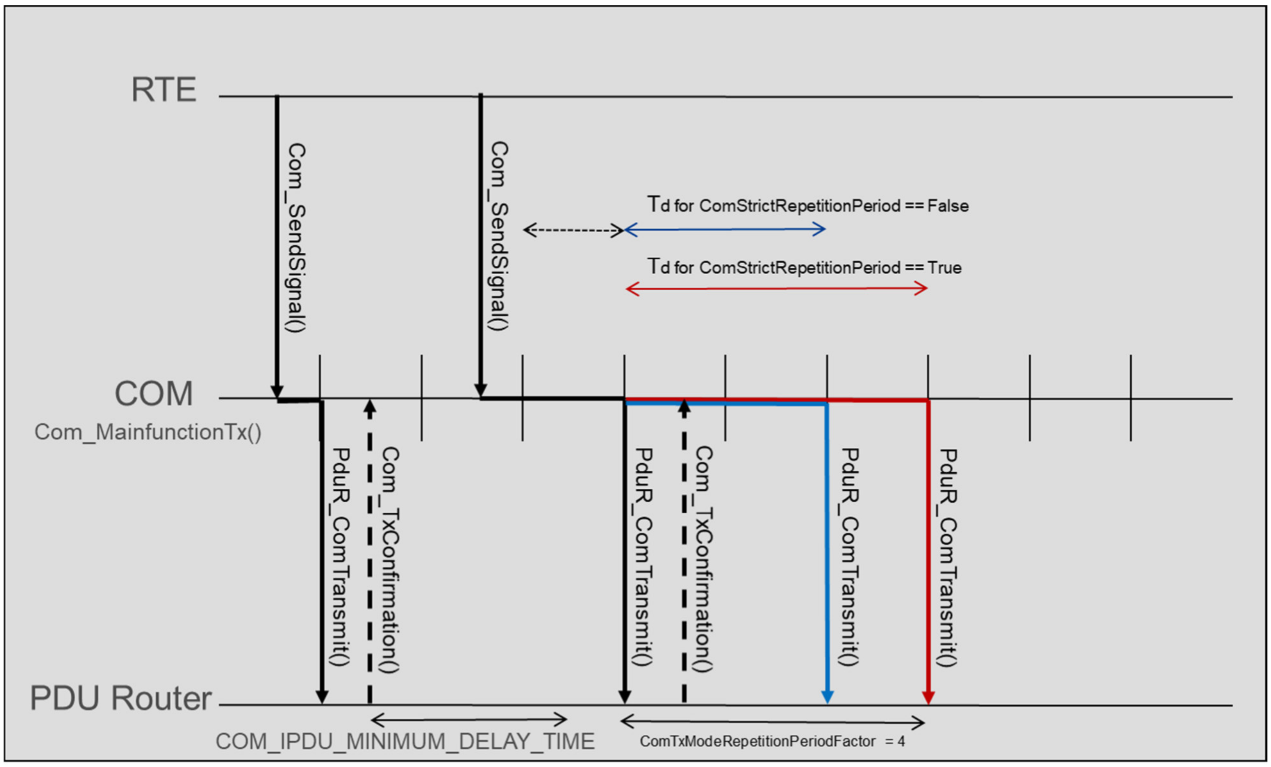

Figure 2-11 Shift of the repetition period caused by the MDT

2.1.28.6 Mixed Mode Periodic Suppression

In the mixed transmission mode periodic and direct transmissions can overlap. By activating the general switch ComMixedModePeriodicSuppression the transmission of the periodic part is suspended following a direct trigger. Once the direct transmission including repetitions has been completed the periodic part is resumed.

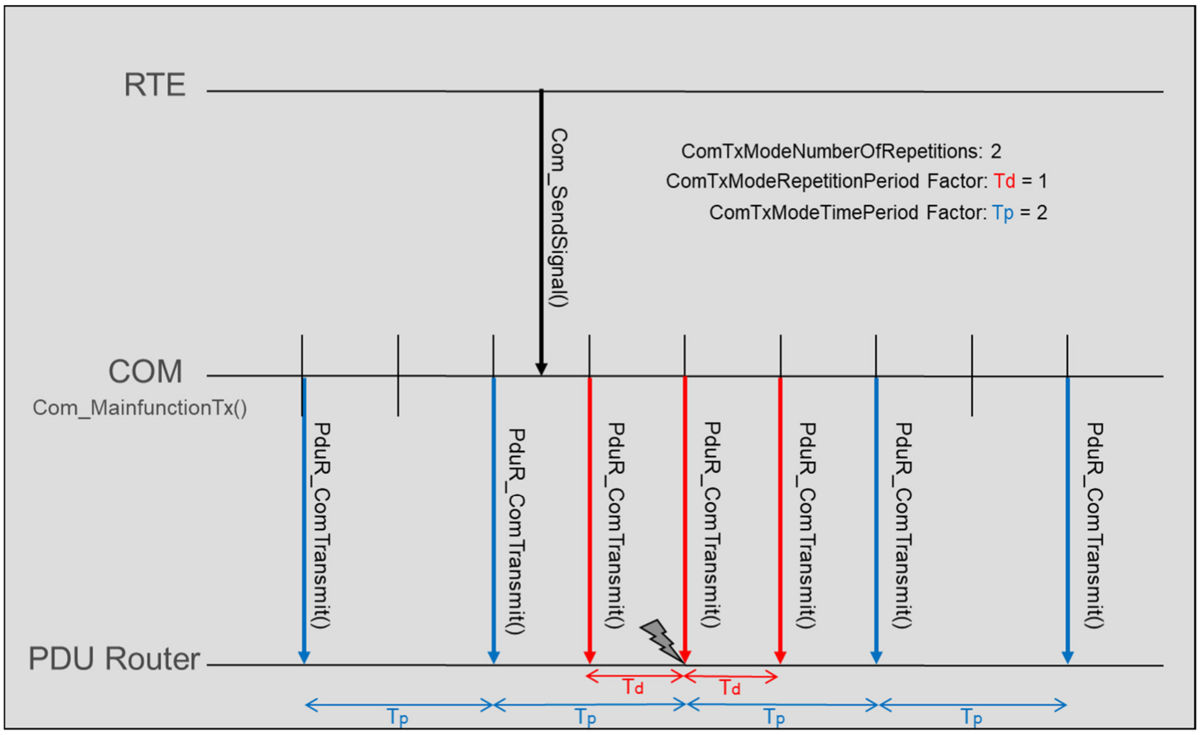

Figure 2-12 Suppression of the periodic part

2.1.29 Main Functions

The configuration of multiple Rx and Tx Main Functions is supported with each Main Function having its own timebase. The names of the Rx an Tx Main Functions depend on the shortname of the respective container in the configuration. There is one Com_MainFunctionRx_<shortname> for each configured ComMainFunctionRx container and one Com_MainFunctionTx_<shortname> for each configured ComMainFunctionTx container. Only one MainFunctionRouteSignals is supported.

In the respective Com_MainFunctionRx_<shortname> only the Rx-IPDUs referencing this Com_MainFunctionrRx_<shortname> are processed. An Rx I-PDU can have a reference to a Com_MainFunctionRx. This reference is necessary if the I-PDU is processed in a main function (e.g. it is a DEFERRED PDU or deadline monitoring is active). In the respective Com_MainFunctionTx_<shortname> only the Tx-IPDUs referencing this Com_MainFunctionTx_<shortname> are processed. A Tx I-PDU must have a reference to a Com_MainFunctionTx.

The timebases should meet the call cycle periods of the main functions respectively. All timing operations are derived from the resolution of these timebases. The Rx/Tx-deadline monitoring operations will rely on the resolution of the timebases, therefore, the configured timeout and first timeout parameters must be a multiple of this resolution. Additionally, if the gateway functionality is configured, the I-PDU-based gateway routing timeout parameter must also meet the above-mentioned requirements. The following cyclic operations will rely on the resolution of the TxMainFunction timebase and therefore have to be a multiple of the configured timebase resolution too:

- Tx Mode Repetition Period

- Tx Mode Time Offset

- Tx Mode Time Period

- Minimum Delay Time

The configured resolution of a timebase defines the granularity, how precisely a timing event is measured. Figure 2-13 shows an example on how precisely a timing event is measured. For example, the reception of a signal should be monitored with a configured timeout of 3ms. An Rx-Event occurs and afterwards the timeout counter is decreased and checked every single tick of the Rx-Timebase until the timeout notification function is called. In Figure 2-13a) a timebase with a resolution of t_{res} = 1ms is used. The timeout counter is decreased 3 times until the Rx-deadline monitoring notification is called. In Figure 2-13b) a timebase with a resolution of t_{res} = 3ms is shown; therefore the timeout counter is decreased and checked less frequently until the timeout notification function is called. However, a coarse resolution can lead to a timeline fuzziness which is shown in Figure 2-13c). Here, the timeline is configured similar to Figure 3-11 b), but this time the n Rx-Event is received right after one clock cycle of the timebase. The timeout counter is not decreased until the next RxMainFunction call, which leads to a worst-case inaccuracy of t_{fuzz} = t_{res}.

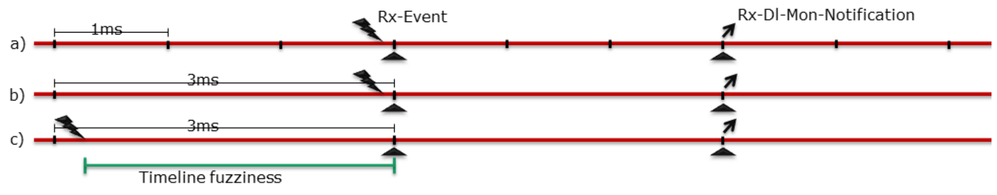

Figure 2-13 Main Function timebase resolution

The old timing domain feature can be mapped to this multiple main functions. Main Functions for Rx-/Tx-deadline monitoring and Tx-cyclic operations can be configured with the time base which was used before for the Rx-/Tx-deadline monitoring timebase or Tx-cyclic timebase. The I-PDUs which perform the deadline monitoring or the cyclic operations have to be mapped to the respective main functions.

2.2 Initialization

Before the COM layer can be used it has to be initialized by Com_Init(). This function performs the basic initialization but does not enable the transmission or reception of signals and I-PDUs.

Initialization, starting and stopping of the layer and its I-PDU groups is normally driven by the Communication Manager. If this software component is not available a similar component has to be provided by the integrator.

If variables exist, which cannot be initialized by the startup code, the function Com_InitMemory() has to be called.

For the multipartition support, the COM shares all RAM between partitions, due to that the Com_Init() function must only be called once to initialize the COM variables.

2.3 States

2.3.1 Module States

The COM module has the following module states

- COM_UNINIT
  The module is not initialized or not usable.
- COM_INIT
  The module is initialized and usable.

and the possible state changes are shown in Figure 2-14.

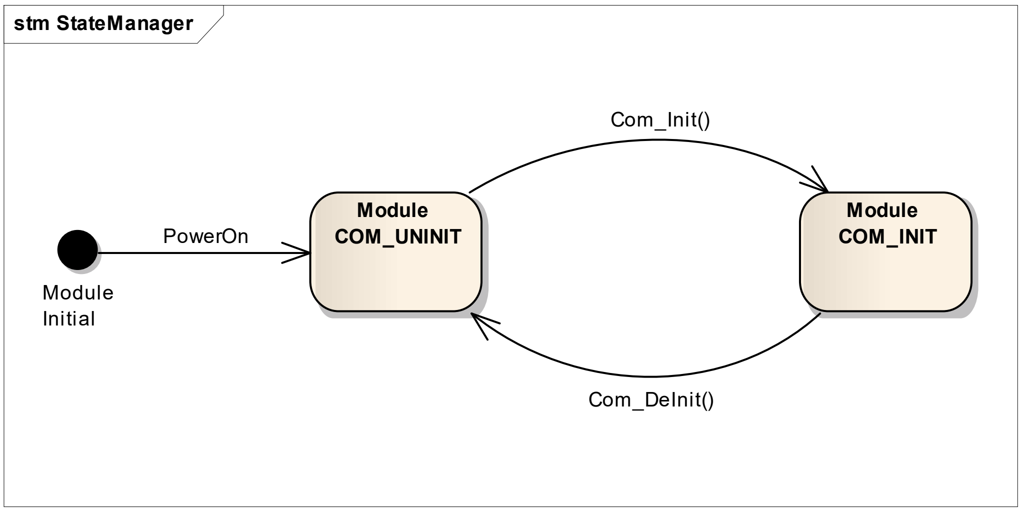

Figure 2-14 Module State Machine

The current state of the COM can be accessed by the API ‘Com_GetStatus()’.

2.3.2 I-PDU States

Each I-PDU has the following states

- Activated
- Deactivated

An I-PDU is active if and only if at least one I-PDU group is active it belongs to. Thus an I-PDU must belong to at least one I-PDU group in order to be able to get activated.

An I-PDU is deactivated if all I-PDU groups it belongs to are deactivated.

I-PDU's without an assigned I-PDU Group are considered to be always active. The state of these I-PDU's cannot be controlled by Com_IPduGroupStart/ Stop APIs.

2.3.3 Reception Deadline Monitoring States

The reception deadline monitoring of an I-PDU is enabled if and only if it is contained in an I-PDU group that has reception deadline monitoring enabled. Otherwise, the reception deadline monitoring of the I-PDU is disabled.

The possible state changes are shown in Figure 2-15.

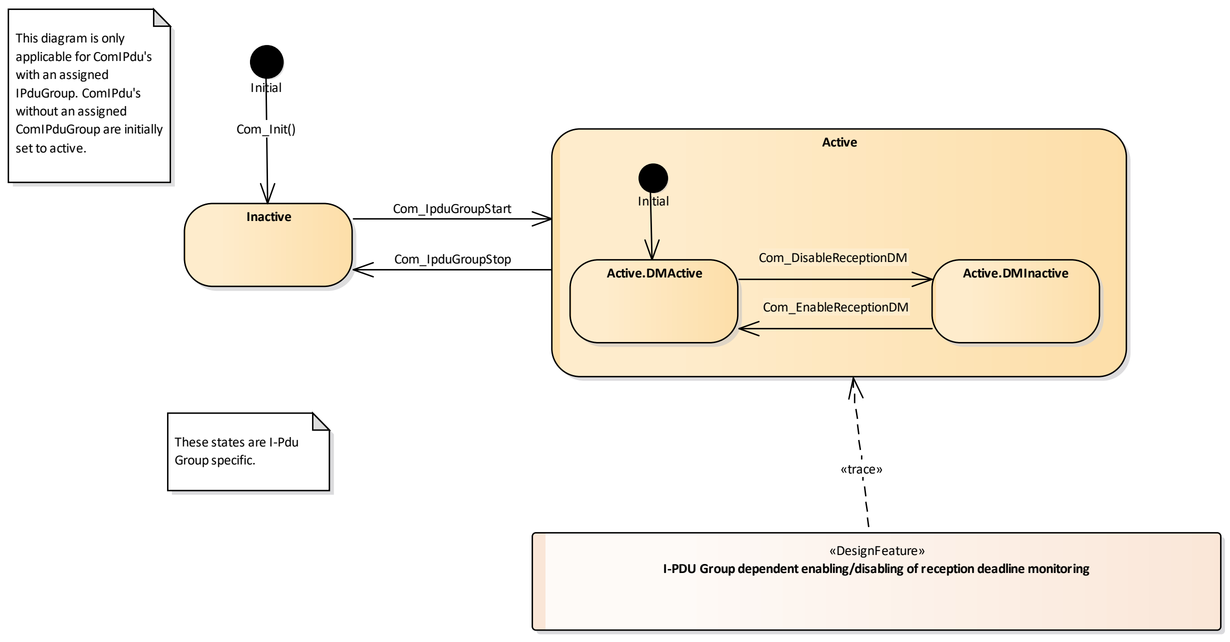
This diagram is only applicable for ComIPdu's with an assigned IPduGroup. ComIPdu's without an assigned ComIPduGroup are initially set to active.
Initial
Com_Init()
Inactive
Com_IpduGroupStart
Com_IpduGroupStop
Active
Initial
Active.DMActive
Com_DisableReceptionDM
Com_EnableReceptionDM
Active.DMInactive
These states are I-Pdu Group specific.
«trace»
«DesignFeature»
I-PDU Group dependent enabling/disabling of reception deadline monitoring

Figure 2-15 Reception Deadline Monitoring State Machine

I-PDU's without an assigned I-PDU Group are considered to be always active, therefore in contrast to Figure 2-15 reception deadline monitoring is initially enabled for Rx I-PDU's as long as at least one timeout is configured for it's contained signal or signal group. The reception deadline monitoring state of these I-PDU's cannot be controlled by Com_EnableReceptionDM/ Com_DisableReceptionDM APIs.

2.4 Main Functions

COM provides following functions listed in Table 2-5 that have to be called cyclically by the Basic Software Scheduler or a similar component.

Main Function	Description
Com_MainFunctionRx_<shortname>()	This function performs the following reception processings > Reception deadline monitoring > Deferred signal processing This function must be called cyclically with a cycle time identical to the configured Rx Time Base.
Com_MainFunctionTx_<shortname>()	This function performs the following transmission processings > Transmission of I-PDUs > Transmission deadline monitoring > Deferred transmission notification This function must be called cyclically with a cycle time identical to the configured Tx Time Base.
Com_MainFunctionRouteSignals_<shortname>()	This function performs the signal gateway functionality. Note that the transmission of Tx I-PDUs is never triggered by this function directly. Thus the Com_MainFunctionTx_<shortname>() is necessary for a complete signal routing. This function must be called cyclically with a cycle time identical to the configured Gw Time Base.

Table 2-5 Main functions that have to be called cyclically

Note
To reduce the signal gateway latency, the order of the main function calls should be as follows

> Com_MainFunctionRx_<shortname>()
> Com_MainFunctionRouteSignals_<shortname>()
> Com_MainFunctionTx_<shortname>()

2.5 Error Handling

2.5.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [2], if development error reporting is enabled (i.e. pre-compile parameter `COM_DEV_ERROR_REPORT==STD_ON`).

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

The reported COM ID is 50.

The reported service IDs identify the services which are described in 0. The following table presents the service IDs and the related services:

Service ID	Service
COMServiceId_Init	Com_Init
COMServiceId_DeInit	Com_DeInit
COMServiceId_IpduGroupStart	Com_ IpduGroupStart
COMServiceId_IpduGroupStop	Com_ IpduGroupStart
COMServiceId_EnableReceptionDM	Com_ EnableReceptionDM
COMServiceId_DisableReceptionDM	Com_ DisableReceptionDM
COMServiceId_GetStatus	Com_GetStatus
COMServiceId_GetConfigurationId	Com_GetConfigurationId
COMServiceId_GetVersionInfo	Com_GetVersionInfo
COMServiceId_SendSignal	Com_SendSignal
COMServiceId_ReceiveSignal	Com_ReceiveSignal
COMServiceId_UpdateShadowSignal	Com_UpdateShadowSignal
COMServiceId_SendSignalGroup	Com_SendSignalGroup
COMServiceId_ReceiveSignalGroup	Com_ReceiveSignalGroup
COMServiceId_ReceiveShadowSignal	Com_ReceiveShadowSignal
COMServiceId_InvalidateSignal	Com_InvalidateSignal
COMServiceId_InvalidateShadowSignal	Com_InvalidateShadowSignal
COMServiceId_TriggerIPDUSend	Com_TriggerIPDUSend
COMServiceId_MainFunctionRx	Com_MainFunctionRx
COMServiceId_MainFunctionTx	Com_MainFunctionTx
COMServiceId_MainFunctionRouteSignals	Com_MainFunctionRouteSignals
COMServiceId_InvalidateSignalGroup	Com_InvalidateSignalGroup
COMServiceId_SendDynSignal	Com_SendDynSignal
COMServiceId_ReceiveDynSignal	Com_ReceiveDynSignal
COMServiceId_SendSignalGroupArray	Com_SendSignalGroupArray
COMServiceId_ReceiveSignalGroupArray	Com_ReceiveSignalGroupArray
COMServiceId_SwitchIpduTxMode	Com_SwitchIpduTxMode

Service ID	Service
COMServiceId_TriggerIPDUSendWithMetaData	Com_TriggerIPDUSendWithMetaData
COMServiceId_TxConfirmation	Com_TxConfirmation
COMServiceId_TriggerTransmit	Com_TriggerTransmit
COMServiceId_RxIndication	Com_RxIndication
COMServiceId_CopyTxData	Com_CopyTxData
COMServiceId_CopyRxData	Com_CopyRxData
COMServiceId_TpRxIndication	Com_TpRxIndication
COMServiceId_StartOfReception	Com_StartOfReception
COMServiceId_TpTxConfirmation	Com_TpTxConfirmation

Table 2-6 Service IDs

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

Error Code	Description
COM_E_PARAM	The API service has been with a wrong parameter.
COM_E_UNINIT	The API service has been called before COM was initialized with Com_Init() or after a call to Com_DeInit()
COM_E_PARAM_POINTER	The API service has been called with a not expected NULL pointer.
COM_E_INIT_FAILED	The API service has been called with a not expected NULL pointer

Table 2-7 Errors reported to DET

2.5.2 Production Code Error Reporting

No production error codes are currently defined for COM.

2.6 Multipartition

In order to provide a load distribution amongst different partitions (cores), the main threads of execution in the COM module, namely the respective MainFunctions are assigned to different partitions. This way the flow of reception/transmission stays within the scope of a single partition.

2.6.1 General configuration

The general concept for the multipartition support of the COM is, that the ComIPdus are processed in the MainFunction which is referenced via the ComIPduMainFunctionRef. The Main Functions are assigned to a partition via the ComMainTx/Rx/RouteSignalsPartitionRef. Additionally, the ComIPdus are assigned to a partition via the “global” Pdu.

Note
The “global” Pdu referenced by the ComIPdu and the MainFunction the ComIPdu references must refer to the same partition.

2.6.2 Initialization and Deinitialization

The COM shares all RAM between partitions, due to that the Com_Init() and Com_DeInit() functions must only be called once to initialize/deinitialize the COM variables.

2.6.3 Restrictions

The following restrictions apply for the Com Multipartition feature:

- The IPduGroups must only contain ComIPdus from one partition.
- Only one Com_MainFunctionRouteSignals per partition is supported.
- TriggerTransmit should only be used “partition local”.
- The following functions should only be used “partition local” or by the master partition (e.g. by BswM): Com_IpduGroupStart, Com_IpduGroupStop, Com_EnableReceptionDM, Com_DisableReceptionDM, Com_SwitchIpduTxMode.
- The Direct Com Invocation of the IpduM can only be used “partition local”. The Pdus of the IpduM must be assigned to the same partition, where the according network type is located at. If more than one network type and thus partition is used, the IpduM needs to support multicore or the Direct Com Invocation can’t be used.
- Immediate Signal Processing is not supported for cross partition Description Routings.

3 Integration

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

3.1 Embedded Implementation

The delivery of the COM contains the files which are described in the chapters 3.1.1 and 3.1.2.

3.1.1 Static Files

File Name	Description
Com_Unity.c	This is the source file of the COM to be used by the compiler. Unit specific c files are included here.
Com.c	This is the source file of the COM. Exclude this file from your build system. The file is build via Com_Unity.c.
Com.h	This is the header file of the COM.
Com_Caching.c	This is the source file of the COM Caching Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_Caching.h	This is the header file of the COM Caching Unit.
Com_CheckUpdateBit.c	This is the source file of the COM Reception Update Bit Checking Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_CheckUpdateBit.h	This is the header file of the COM Reception Update Bit Checking Unit.
Com_CprQueue.c	This is the source file of the COM Cross Partition Routing Queue Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_CprQueue.h	This is the header file of the COM Cross Partition Routing Queue Unit.
Com_CprQueueDescrGw.c	This is the source file of the COM Cross Partition Description Routing Queue Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_CprQueueDescrGw.h	This is the header file of the COM Cross Partition Description Routing Queue Unit.
Com_CprQueueSigGw.c	This is the source file of the COM Cross Partition Signal Routing Queue Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_CprQueueSigGw.h	This is the header file of the COM Cross Partition Signal Routing Queue Unit.
Com_Deserializer.c	This is the source file of the COM Deserializer Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_Deserializer.h	This is the header file of the COM Deserializer Unit.

File Name	Description
Com_DesGw.c	This is the source file of the COM Description Routing Gateway Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_DesGw.h	This is the header file of the COM Description Routing Gateway Unit.
Com_DesGwBc.c	This is the source file of the COM Description Gateway Bit Copy Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_DesGwBc.h	This is the header file of the COM Description Gateway Bit Copy Unit.
Com_DesGwCp.c	This is the source file of the COM Description Gateway Cross Partition Routing Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_DesGwCp.h	This is the header file of the COM Description Gateway Cross Partition Routing Unit.
Com_DesGwSp.c	This is the source file of the COM Description Gateway Single Partition Routing Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_DesGwSp.h	This is the header file of the COM Description Gateway Single Partition Routing Unit.
Com_EventCache.c	This is the source file of the COM Event Cache Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_EventCache.h	This is the header file of the COM Event Cache Unit.
Com_GwTout.c	This is the source file of the COM Gateway Timeout Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_GwTout.h	This is the header file of the COM Gateway Timeout Unit.
Com_Initalization.c	This is the source file of the COM Initalization Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_Initalization.h	This is the header file of the COM Initalization Unit.
Com_IPduGroupHdlr.c	This is the source file of the COM IPduGroup Handler Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_IPduGroupHdlr.h	This is the header file of the COM IPduGroup Handler Unit.
Com_ISRThreshold.c	This is the source file of the COM ISRThreshold Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_ISRThreshold.h	This is the header file of the COM ISRThreshold Unit.
Com_LLRxIf.c	This is the source file of the COM Lower Layer Reception Interface Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_LLRxIf.h	This is the header file of the COM Lower Layer Reception Interface Unit.

File Name	Description
Com_LLRxTp.c	This is the source file of the COM Lower Layer Reception Transport Protocol Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_LLRxTp.h	This is the header file of the COM Lower Layer Reception Transport Protocol Unit.
Com_LLTxIf.c	This is the source file of the COM Layer Transmission Interface Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_LLTxIf.h	This is the header file of the COM Layer Transmission Interface Unit.
Com_LLTxTp.c	This is the source file of the COM Lower Layer Transmission Transport Protocol Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_LLTxTp.h	This is the header file of the COM Lower Layer Transmission Transport Protocol Unit.
Com_MainFunctions.c	This is the source file of the COM MainFunctions Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_MainFunctions.h	This is the header file of the COM MainFunctions Unit.
Com_Notifications.c	This is the source file of the COM Notifications Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_Notifications.h	This is the header file of the COM Notifications Unit.
Com_Repetition.c	This is the source file of the COM Repetition Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_Repetition.h	This is the header file of the COM Repetition Unit.
Com_Reporting.c	This is the source file of the COM Reporing Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_Reporting.h	This is the header file of the COM Reporting Unit.
Com_RxDlMon.c	This is the source file of the COM Reception Deadline Monitoring Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_RxDlMon.h	This is the header file of the COM Reception Deadline Monitoring Unit.
Com_RxInv.c	This is the source file of the COM Rx Invalidation Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_RxInv.h	This is the header file of the COM Rx Invalidation Unit.
Com_RxPduBuffer.c	This is the source file of the COM Reception Pdu Buffer Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_RxPduBuffer.h	This is the header file of the COM Reception Pdu Buffer Unit.

File Name	Description
Com_RxPduProcessing.c	This is the source file of the COM Rx Pdu Processing Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_RxPduProcessing.h	This is the header file of the COM Rx Pdu Processing Unit.
Com_RxSigBuffer.c	This is the source file of the COM Reception Signal Buffer Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_RxSigBuffer.h	This is the header file of the COM Reception Signal Buffer Unit.
Com_RxSigBufferHelper.c	This is the source file of the COM Reception Signal Buffer Helper Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_RxSigBufferHelper.h	This is the header file of the COM Reception Signal Buffer Helper Unit.
Com_RxSignalFiltering.c	This is the source file of the COM Reception Filtering Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_RxSignalFiltering.h	This is the header file of the COM Reception Filtering Unit.
Com_RxSignalIf.c	This is the source file of the COM Reception Signal Interface Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_RxSignalIf.h	This is the header file of the COM Reception Signal Interface Unit.
Com_RxSignalProcessing.c	This is the source file of the COM Reception Signal Processing Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_RxSignalProcessing.h	This is the header file of the COM Reception Signal Processing Unit.
Com_Serializer.c	This is the source file of the COM Serializer Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_Serializer.h	This is the header file of the COM Serializer Unit.
Com_SigGw.c	This is the source file of the COM Signal Routing Gateway Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_SigGw.h	This is the header file of the COM Signal Routing Gateway Unit.
Com_SigGwCP.c	This is the source file of the COM Signal Gateway Cross Partition Routing Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_SigGwCP.h	This is the header file of the COM Signal Gateway Cross Partition Routing Unit.
Com_SigGwSP.c	This is the source file of the COM Signal Gateway Single Partition Routing Unit. Exclude this file from your build system. The file is build via Com_Unity.c.

File Name	Description
Com_SigGwSP.h	This is the source file of the COM Signal Gateway Single Partition Routing Unit.
Com_SignalFilterHdlr.c	This is the source file of the COM Signal Filter Handler Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_SignalFilterHdlr.h	This is the header file of the COM Signal Filter Handler Unit.
Com_SignalFilterHdlrHelper.c	This is the source file of the COM Signal Filter Handler Helper Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_SignalFilterHdlrHelper.h	This is the header file of the COM Signal Filter Handler Helper Unit.
Com_Timer.c	This is the source file of the COM Timer Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_Timer.h	This is the header file of the COM Timer Unit.
Com_TxBuffer.c	This is the source file of the COM Tx Pdu Buffer Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_TxBuffer.h	This is the header file of the COM Tx Pdu Buffer Unit.
Com_TxCyclic.c	This is the source file of the COM Cyclic Transmssion Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_TxCyclic.h	This is the header file of the COM Cyclic Transmssion Unit.
Com_TxDlMon.c	This is the source file of the COM Transmission Deadline Monitoring Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_TxDlMon.h	This is the header file of the COM Transmission Deadline Monitoring Unit.
Com_TxGroupSignalProcessing.c	This is the source file of the COM Transmission Group Signal Processing Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_TxGroupSignalProcessing.h	This is the header file of the COM Transmission Group Signal Processing Unit.
Com_TxInv.c	This is the source file of the COM Tx Invalidation Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_TxInv.h	This is the header file of the COM Tx Invalidation Unit.
Com_TxMinDelay.c	This is the source file of the COM Transmission Minimum Delay Time Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_TxMinDelay.h	This is the header file of the COM Transmission Minimum Delay Time Unit.
Com_TxModeHdlr.c	This is the source file of the COM Tx Mode Handler Unit. Exclude this file from your build system. The file is build via Com_Unity.c.

File Name	Description
Com_TxModeHdlr.h	This is the header file of the COM Tx Mode Handler Unit.
Com_TxSignalFiltering.c	This is the source file of the COM Transmission Signal Filtering Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_TxSignalFiltering.h	This is the header file of the COM Transmission Signal Filtering Unit.
Com_TxSignalIf.c	This is the source file of the COM Transmission Signal Interface Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_TxSignalIf.h	This is the header file of the COM Transmission Signal Interface Unit.
Com_TxSignalProcessing.c	This is the source file of the COM Transmission Signal Processing Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_TxSignalProcessing.h	This is the header file of the COM Transmission Signal Processing Unit.
Com_TxTransmit.c	This is the source file of the COM Transmission Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_TxTransmit.h	This is the header file of the COM Transmission Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_Util.c	This is the source file of the COM Util Unit. Exclude this file from your build system. The file is build via Com_Unity.c.
Com_Util.h	This is the header file of the COM Util Unit.

Table 3-1 Static files

3.1.2 Dynamic Files

The dynamic files are generated by the configuration tool DaVinci Configurator.

File Name	Description–
Com_Cfg.h / Com_Cfg_<x>.h	This file contains: > global constant macros > global function macros > global data types and structures > global data prototypes > global function prototypes of CONFIG-CLASS PRE-COMPILE data.
Com_Cbk.h	This is the generated header file of COM containing prototypes for lower layers.
Com_Cfg.c	This file contains: > local constant macros > local function macros

File Name	Description–
	> local data types and structures > local data prototypes > local data > global data of CONFIG-CLASS PRE-COMPILE data.
Com_Cot.h	This file contains the prototypes of: > I-Pdu Callouts > I-Pdu Trigger Transmit Callouts This file is included by Com_Cbk.h.
Com_Lcfg.h	This file contains: > global constant macros > global function macros > global data types and structures > global data prototypes > global function prototypes of CONFIG-CLASS LINK data.
Com_Lcfg.c	This file contains: > local constant macros > local function macros > local data types and structures > local data prototypes > local data > global data of CONFIG-CLASS LINK data.
Com_PBcfg.h	This file contains: > global constant macros > global function macros > global data types and structures > global data prototypes > global function prototypes of CONFIG-CLASS POST-BUILD data.
Com_PBcfg.c	This file contains: > local constant macros > local function macros > local data types and structures > local data prototypes > local data > global data of CONFIG-CLASS POST-BUILD data.
Com_Types.h	This file contains the static type definitions.

Table 3-2 Generated files

3.2 Critical Sections

The handling of entering and leaving critical sections is provided by the RTE. The Vector MICROSAR Classic COM offers several groups of critical sections to optimize the runtime consumption.

For a more detailed description of the possible exclusive area implementation variant please refer to [4].

Following critical sections are offered

- COM_EXCLUSIVE_AREA_BOTH

  This critical section protects Rx and Tx resources that are being accessed in context of Com_MainFunctionRouteSignals for signal gateway routings or description routings with configured deferred description source. Therefore the critical section enclosed with COM_EXCLUSIVE_AREA_BOTH should never be interrupted by any Com API which accesses Tx or Rx resources.

- COM_EXCLUSIVE_AREA_TX

  This critical section protects Tx resources that can be accessed from various contexts. Therefore the critical section enclosed with COM_EXCLUSIVE_AREA_TX should never be interrupted by any Com API which accesses Tx resources.

- COM_EXCLUSIVE_AREA_RX

  This critical section protects Rx resources that can be accessed from various contexts. Therefore the critical section enclosed with COM_EXCLUSIVE_AREA_RX should never be interrupted by any Com API which accesses Rx resources.

Note
See Com.h for documentation of the critical sections used in each Com API function.

4 API Description

For an interfaces overview please see Figure 1-1.

4.1 Type Definitions

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

Type Name	C-Type	Description	Value Range
Com_StatusType	enum	This is a status value returned by the API service Com_GetStatus().	`COM_UNINIT` The AUTOSAR COM module is not initialized or not usable
			`COM_INIT` The AUTOSAR COM Module is initialized and usable.
Com_SignalIdType	c-type	AUTOSAR COM signal object identifier.	0..<SignalIdmax> Zero-based integer number
Com_SignalGroupIdType	c-type	AUTOSAR COM signal group object identifier.	0..<SignalGroupIdmax> Zero-based integer number
Com_IpduGroupIdType	c-type	AUTOSAR COM I-PDU group object identifier.	0..<IpduGroupIdmax> Zero-based integer number
Com_SerciveIdType	enum	Unique identifier of an AUTOSAR COM service.	See in Table 2-6

Table 4-1 Type definitions

4.2 Services provided by COM

4.2.1 Com_Init

Prototype	
`void Com_Init (const Com_ConfigType *config)`	
Parameter	
config [in]	NULL_PTR if COM_USE_INIT_POINTER is STD_OFF Pointer to the Com configuration data if COM_USE_INIT_POINTER is STD_ON
Return code	
void	none
Functional Description	
This service initializes internal and external interfaces and variables of the AUTOSAR COM layer for the further processing. After calling this function the inter-ECU communication is still disabled.	
Particularities and Limitations	
> Com_InitMemory() has to be executed previously, if the startup code does not initialize variables.Com is not in initialized state.	
!	**Caution** 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.
Call context	
> The function must be called on task level and must not be interrupted by other administrative function calls. > This function is Synchronous	

Table 4-2 Com_Init

4.2.2 Com_InitMemory

Prototype	
`void Com_InitMemory (void)`	
Parameter	
void	none
Return code	
void	none
Functional Description	
The function initializes variables, which cannot be initialized with the startup code.	
Particularities and Limitations	
Com_Init() is not called yet.	
Call context	
> The function must be called on task level.	

Table 4-3 Com_InitMemory

4.2.3 Com_DeInit

Prototype	
`void Com_DeInit (void)`	
Parameter	
void	none
Return code	
void	none
Functional Description	
This service stops the inter-ECU communication. All started I-PDU groups 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.	
Particularities and Limitations	
-	
❗	**Caution** 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.
Call context	
> The function must be called on task level and must not be interrupted by other administrative function calls. > This function is Synchronous	

Table 4-4 Com_DeInit

4.2.4 Com_IpduGroupStart

Prototype	
`void Com_IpduGroupStart (Com_IpduGroupIdType IpduGroupId, boolean Initialize)`	
Parameter	
IpduGroupId [in]	ID of I-PDU group to be started
Initialize [in]	Flag to request initialization of the data in the I-PDUs of this I-PDU group
Return code	
void	none
Functional Description	
Starts a preconfigured I-PDU group. For example, cyclic I-PDUs will be sent out cyclically after the call of Com_IpduGroupStart(). If Initialize is true all I-PDUs of the I-PDU group shall be (re-)initialized before the I-PDU group is started. That means 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.	
Particularities and Limitations	
-	

❗	**Caution** A call to Com_IpduGroupStart shall not be interrupted by another call to Com_IpduGroupStart, Com_EnableReceptionDM, Com_DisableReceptionDM or a call to Com_IpduGroupStop.
Call context	
> The function must be called on task level and must not be interrupted by other Com_IpduGroupStart and Com_IpduGroupStop calls. > This function is Synchronous	

Table 4-5 Com_IpduGroupStart

4.2.5 Com_IpduGroupStop

Prototype	
`void Com_IpduGroupStop (Com_IpduGroupIdType IpduGroupId)`	
Parameter	
IpduGroupId [in]	ID of I-PDU group to be stopped
Return code	
void	none
Functional Description	
Stops a preconfigured I-PDU group. For example, cyclic I-PDUs will be stopped after the call of Com_IpduGroupStop().	
Particularities and Limitations	
-	
❗	**Caution** A call to Com_IpduGroupStop shall not be interrupted by another call to Com_IpduGroupStop, Com_EnableReceptionDM, Com_DisableReceptionDM or a call to Com_IpduGroupStart.
Call context	
> The function must be called on task level. > This function is Synchronous	

Table 4-6 Com_IpduGroupStop

4.2.6 Com_EnableReceptionDM

Prototype	
`void Com_EnableReceptionDM (Com_IpduGroupIdType IpduGroupId)`	
Parameter	
IpduGroupId [in]	ID of I-PDU group where reception DM shall be enabled.
Return code	
void	none

Functional Description	
Enables the reception deadline monitoring for the I-PDUs within the given I-PDU group. This call has no effect if the corresponding I-PDU group is disabled.	
Particularities and Limitations	
-	
!	**Caution** A call to Com_EnableReceptionDM shall not be interrupted by another call to Com_EnableReceptionDM, Com_IpduGroupStop, Com_DisableReceptionDM or a call to Com_IpduGroupStart.
Call context	
> The function must be called on task level. > This function is Synchronous	

Table 4-7 Com_EnableReceptionDM

4.2.7 Com_DisableReceptionDM

Prototype	
`void` **`Com_DisableReceptionDM`** `(Com_IpduGroupIdType IpduGroupId)`	
Parameter	
IpduGroupId [in]	ID of I-PDU group where reception DM shall be disabled.
Return code	
void	none
Functional Description	
Disables the reception deadline monitoring for the I-PDUs within the given I-PDU group.	
Particularities and Limitations	
-	
!	**Caution** A call to Com_DisableReceptionDM shall not be interrupted by another call to Com_DisableReceptionDM, Com_IpduGroupStop, Com_EnableReceptionDM or a call to Com_IpduGroupStart.
Call context	
> The function must be called on task level. > This function is Synchronous	

Table 4-8 Com_DisableReceptionDM

4.2.8 Com_GetConfigurationId

Prototype	
`uint32 Com_GetConfigurationId (void)`	
Parameter	
void	none
Return code	
uint32	none
	none
	uint32 Configured ConfigurationID
Functional Description	
This function shall return the unique identifier of the configuration.	
Particularities and Limitations	
-	
Call context	
> The function can be called on interrupt and task level. CREQ-107420 > This function is Synchronous	

Table 4-9 Com_GetConfigurationId

4.2.9 Com_GetStatus

Prototype	
`Com_StatusType Com_GetStatus (void)`	
Parameter	
void	none
Return code	
Com_StatusType	Com_StatusType
Functional Description	
Returns the status of the AUTOSAR COM module.	
Particularities and Limitations	
-	
Call context	
> The function can be called on interrupt and task level. CREQ-107163 > This function is Synchronous	

Table 4-10 Com_GetStatus

4.2.10 Com_GetVersionInfo

Prototype	
`void Com_GetVersionInfo (Std_VersionInfoType *versioninfo)`	
Parameter	
versioninfo [out]	Pointer to where to store the version information of this module.
Return code	
void	none
Functional Description	
Returns the version information of this module.	
Particularities and Limitations	
-	
Call context	
> The function can be called on interrupt and task level. > This function is Synchronous	

Table 4-11 Com_GetVersionInfo

4.2.11 Com_TriggerIPDUSend

Prototype	
`void Com_TriggerIPDUSend (PduIdType PduId)`	
Parameter	
PduId [in]	ID of Tx I-PDU.
Return code	
void	void
Functional Description	
By a call to Com_TriggerIPDUSend the I-PDU with the given ID is triggered for transmission.	
Particularities and Limitations	
-	
Call context	
> The function can be called on interrupt and task level. > This function is Synchronous	

Table 4-12 Com_TriggerIPDUSend

4.2.12 Com_TriggerIPDUSendWithMetaData

Prototype
`void Com_TriggerIPDUSendWithMetaData (PduIdType PduId, const uint8 *MetaData)`

Parameter	
PduId [in]	ID of Tx I-PDU.
MetaData [in]	The Meta data that shall be added to the I-PDU before sending.
Return code	
void	void
Functional Description	
By a call to Com_TriggerIPDUSendWithMetaData the given meta data is appended to the I-PDU and the I-PDU with the given ID is triggered for transmission.	
Particularities and Limitations	
-	
Call context	
> The function can be called on interrupt and task level. > This function is Synchronous	

Table 4-13 Com_TriggerIPDUSendWithMetaData

4.2.13 Com_ReceiveDynSignal

Prototype	
`uint8 Com_ReceiveDynSignal (Com_SignalIdType SignalId, void *SignalDataPtr, uint16 *Length)`	
Parameter	
SignalId [in]	Id of signal or group signal to be received.
SignalDataPtr [out]	Reference to the signal data in which to store the received data.
Length [in, out]	in: maximum length that could be received out: length of the dynamic length signal
Return code	
uint8	uint8 E_OK service has been accepted E_NOT_OK in case the Length (as in-parameter) is smaller than the received length of the dynamic length signal COM_SERVICE_NOT_AVAILABLE corresponding I-PDU group was stopped (or service failed due to development error) COM_BUSY in case the TP-Buffer is locked
Functional Description	
The service Com_ReceiveDynSignal updates the signal data referenced by SignalDataPtr with the data in the signal object identified by SignalId. The Length parameter indicates as "in parameter" the maximum length that can be received and as "out parameter" the length of the written dynamic length signal or group signal. If the signal processing of the corresponding I-Pdu is configured to DEFERRED the last received signal value is available not until the next call to the respective Com_MainfunctionRx. If a group signal is read, the data in the shadow buffer should be updated before the call by a call of the service Com_ReceiveSignalGroup.	
Particularities and Limitations	
-	
Call context	
> The function can be called on interrupt and task level.	

> This function is Synchronous

Table 4-14 Com_ReceiveDynSignal

4.2.14 Com_ReceiveSignalGroup

Prototype	
`uint8 Com_ReceiveSignalGroup (Com_SignalGroupIdType SignalGroupId)`	
Parameter	
SignalGroupId [in]	Id of signal group to be received.
Return code	
uint8	uint8 E_OK service has been accepted COM_SERVICE_NOT_AVAILABLE corresponding I-PDU group was stopped (or service failed due to development error)
Functional Description	
The service Com_ReceiveSignalGroup copies the received signal group to the shadow buffer. After this call, the group signals could be copied from the shadow buffer to the upper layer by a call of Com_ReceiveShadowSignal.	
Particularities and Limitations	
-	
Call context	
> The function can be called on interrupt and task level. To guarantee data consistency of the whole signal group the complete reception of a signal group (consecutive calls of 'Com_ReceiveSignalGroup' and 'Com_ReceiveSignal') must not be interrupted by another reception request for the same signal group. > This function is Synchronous	

Table 4-15 Com_ReceiveSignalGroup

4.2.15 Com_ReceiveSignalGroupArray

Prototype	
`uint8 Com_ReceiveSignalGroupArray (Com_SignalGroupIdType SignalGroupId, uint8 *SignalGroupArrayPtr)`	
Parameter	
SignalGroupId [in]	Id of signal group to be received.
SignalGroupArrayPtr [out]	reference to the location where the received signal group array shall be stored
Return code	
uint8	uint8 E_OK service has been accepted COM_SERVICE_NOT_AVAILABLE corresponding I-PDU group was stopped (or service failed due to development error) COM_BUSY in case the TP-Buffer is locked for large data types handling

Functional Description
The service Com_ReceiveSignalGroupArray copies the received signal group array representation from the PDU to the location designated by SignalGroupArrayPtr.
Particularities and Limitations
The configuration switch ComEnableSignalGroupArrayApi has to be enabled.
Call context
> The function can be called on interrupt and task level. > This function is Synchronous

Table 4-16 Com_ReceiveSignalGroupArray

4.2.16 Com_InvalidateSignal

Prototype	
`uint8 Com_InvalidateSignal (Com_SignalIdType SignalId)`	
Parameter	
SignalId [in]	ID of signal or group signal to be invalidated.
Return code	
uint8	uint8 E_OK service has been accepted COM_SERVICE_NOT_AVAILABLE corresponding I-PDU group was stopped (or service failed due to development error)
Functional Description	
This function invalidates the signal or group signal by calling Com_SendSignal with the configured invalid value. If this function is used to invalidate a group signal, a call to Com_SendSignalGroup is needed to update the signal group data.	
Particularities and Limitations	
-	
Call context	
> The function can be called on interrupt and task level and has not to be interrupted by other Com_SendSignal and Com_InvalidateSignal calls for the same SignalId.	

Table 4-17 Com_InvalidateSignal

4.2.17 Com_InvalidateSignalGroup

Prototype	
`uint8 Com_InvalidateSignalGroup (Com_SignalGroupIdType SignalGroupId)`	
Parameter	
SignalGroupId [in]	ID of signal group to be invalidated.
Return code	
uint8	uint8 E_OK service has been accepted COM_SERVICE_NOT_AVAILABLE corresponding I-PDU group was stopped (or service failed due to development error)

Functional Description
This function invalidates the whole signal group by calling Com_SendSignal with the configured invalid value for all group signals of the signal group. After invalidation of the current signal group data Com_SendSignalGroup is performed internally.
Particularities and Limitations
-
Call context
> The function can be called on interrupt and task level and has not to be interrupted by other Com_InvalidateSignalGroup calls for the same SignalGroupId and by Com_SendSignal calls for a SignalId which is contained in the same signal group.

Table 4-18 Com_InvalidateSignalGroup

4.2.18 Com_SwitchIpduTxMode

Prototype	
`void Com_SwitchIpduTxMode (PduIdType PduId, boolean Mode)`	
Parameter	
PduId [in]	ID of Tx I-PDU.
Mode [in]	TX mode of the I-PDU (TRUE/FALSE)
Return code	
void	none
Functional Description	
This method sets the TX Mode of the I-PDU referenced by PduId to Mode. In case the TX Mode changes the new mode is immediately effective. In case the requested transmission mode was already active for this I-PDU, the call will have no effect.	
Particularities and Limitations	
Call context	
> The function can be called on interrupt and task level > This function is Synchronous	

Table 4-19 Com_SwitchIpduTxMode

4.2.19 Com_SendDynSignal

Prototype	
`uint8 Com_SendDynSignal (Com_SignalIdType SignalId, const void *SignalDataPtr, uint16 Length)`	
Parameter	
SignalId [in]	ID of signal or group signal to be sent.
SignalDataPtr [in]	Reference to the signal data to be transmitted.
Length [in]	Length of the dynamic length signal.

Return code	
uint8	uint8 E_OK service has been accepted COM_SERVICE_NOT_AVAILABLE corresponding I-PDU group was stopped (or service failed due to development error) COM_BUSY in case the TP-Buffer is locked for large data types handling
Functional Description	
The service Com_SendDynSignal updates the signal or group signal object identified by SignalId with the signal data referenced by the SignalDataPtr parameter. The Length parameter is evaluated for dynamic length signals.	
Particularities and Limitations	
-	
Call context	
> The function can be called on interrupt and task level and has not to be interrupted by other Com_SendSignal and Com_InvalidateSignal calls for the same SignalId.	

Table 4-20 Com_SendDynSignal

4.2.20 Com_SendSignal

Prototype	
`uint8 Com_SendSignal (Com_SignalIdType SignalId, const void *SignalDataPtr)`	
Parameter	
SignalId [in]	ID of signal or group signal to be sent.
SignalDataPtr [in]	Reference to the signal data to be transmitted.
Return code	
uint8	uint8 E_OK service has been accepted COM_SERVICE_NOT_AVAILABLE corresponding I-PDU group was stopped (or service failed due to development error) COM_BUSY in case the TP-Buffer is locked for large data types handling
Functional Description	
The service Com_SendSignal updates the signal or group signal object identified by SignalId with the signal data referenced by the SignalDataPtr parameter.	
Particularities and Limitations	
-	
Call context	
> The function can be called on interrupt and task level and has not to be interrupted by other Com_SendSignal and Com_InvalidateSignal calls for the same SignalId.	

Table 4-21 Com_SendSignal

4.2.21 Com_SendSignalGroup

Prototype	
`uint8 Com_SendSignalGroup (Com_SignalGroupIdType SignalGroupId)`	
Parameter	
SignalGroupId [in]	ID of signal group to be send.
Return code	
uint8	uint8 E_OK service has been accepted COM_SERVICE_NOT_AVAILABLE corresponding I-PDU group was stopped (or service failed due to development error)
Functional Description	
The service Com_SendSignalGroup copies the content of the associated shadow buffer to the associated I-PDU buffer. Prior to this call, all group signals should be updated in the shadow buffer by the call of Com_SendSignal.	
Particularities and Limitations	
-	
Call context	
> The function can be called on interrupt and task level and has not to be interrupted by other Com_SendSignalGroup calls for the same SignalGroupId. To guarantee data consistency of the whole signal group the complete transmission of a signal group (consecutive calls of 'Com_SendSignal' and 'Com_SendSignalGroup') must not be interrupted by another transmission request for the same signal group or by a call of 'Com_InvalidateSignalGroup'.	

Table 4-22 Com_SendSignalGroup

4.2.22 Com_SendSignalGroupArray

Prototype	
`uint8 Com_SendSignalGroupArray (Com_SignalGroupIdType SignalGroupId, const uint8 *SignalGroupArrayPtr)`	
Parameter	
SignalGroupId [in]	Id of signal group to be sent.
SignalGroupArrayPtr [in]	Reference to the signal group array.
Return code	
uint8	uint8 E_OK service has been accepted COM_SERVICE_NOT_AVAILABLE corresponding I-PDU group was stopped (or service failed due to development error) COM_BUSY in case the TP-Buffer is locked for large data types handling
Functional Description	
The service Com_SendSignalGroupArray copies the content of the provided SignalGroupArrayPtr to the associated I-PDU. The provided data shall correspond to the array representation of the signal group.	
Particularities and Limitations	
The configuration switch ComEnableSignalGroupArrayApi has to be enabled.	
Call context	

> The function can be called on interrupt and task level.

Table 4-23 Com_SendSignalGroupArray

4.2.23 Com_MainFunctionRx

Prototype	
`void Com_MainFunctionRx (void)`	
Parameter	
void	none
Return code	
void	none
Functional Description	
Per configured ComMainFunctionRx instance one Com_MainFunctionRx_<shortname> is implemented. Hereby <shortname> is the short name of the ComMainFunctionRx configuration container in the ECU configuration. 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 returns simply if COM was not previously initialized with a call to Com_Init.	
Particularities and Limitations	
- CREQ-103161	
Call context	
> The function must be called on task level.	

Table 4-24 Com_MainFunctionRx

4.2.24 Com_MainFunctionTx

Prototype	
`void Com_MainFunctionTx (void)`	
Parameter	
void	none
Return code	
void	none
Functional Description	
Per configured ComMainFunctionTx instance one Com_MainFunctionTx_<shortname> is implemented. Hereby <shortname> is the short name of the ComMainFunctionTx configuration container in the ECU configuration. This function shall perform the processing of the transmission activities that are not directly initiated by the calls from the RTE and PDU-R. A call to Com_MainFunctionTx returns simply if COM was not previously initialized with a call to Com_Init.	

Particularities and Limitations
- CREQ-103168
Call context
> The function must be called on task level.

Table 4-25 Com_MainFunctionTx

4.2.25 Com_MainFunctionRouteSignals

Prototype	
`void` **`Com_MainFunctionRouteSignals`** `(void)`	
Parameter	
void	none
Return code	
void	None
Functional Description	
Per configured ComMainFunctionRouteSignals instance one Com_MainFunction RouteSignals_<shortName> is implemented. Hereby <shortName> is the short name of the ComMainFunctionRouteSignals configuration container in the ECU configuration. 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. A call to Com_MainFunctionRouteSignals returns simply if COM was not previously initialized with a call to Com_Init.	
Particularities and Limitations	
-	
!	**Caution** The time between to consecutive calls (perhaps the related task/thread cycle) affects directly the signal gateway latency. CREQ-103192
Call context	
> The function must be called on task level.	

Table 4-26 Com_MainFunctionRouteSignals

4.2.26 Com_ReceiveSignal

Prototype	
`uint8` **`Com_ReceiveSignal`** `(Com_SignalIdType SignalId, void *SignalDataPtr)`	
Parameter	
SignalId [in]	Id of signal or group signal to be received.
SignalDataPtr [out]	Reference to the signal data in which to store the received data.

Return code	
uint8	uint8 E_OK service has been accepted COM_SERVICE_NOT_AVAILABLE corresponding I-PDU group was stopped (or service failed due to development error)
Functional Description	
The service Com_ReceiveSignal updates the signal data referenced by SignalDataPtr with the data in the signal object identified by SignalId. If the signal processing of the corresponding I-Pdu is configured to DEFERRED the last received signal value is available not until the next call to the respective Com_MainfunctionRx. If a group signal is read, the data in the shadow buffer should be updated before the call by a call of the service Com_ReceiveSignalGroup.	
Particularities and Limitations	
-	
Call context	
> The function can be called on interrupt and task level. > This function is Synchronous	

Table 4-27 Com_ReceiveSignal

4.2.27 Com_ReceiveShadowSignal

Prototype	
`uint8 **Com_ReceiveShadowSignal** (Com_SignalIdType SignalId, void *SignalDataPtr)`	
Parameter	
SignalId [in]	Id of group signal to be received.
SignalDataPtr [out]	Reference to the group signal data in which to store the received data.
Return code	
uint8	void
Functional Description	
The service Com_ReceiveShadowSignal updates the group signal data 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.	
Particularities and Limitations	
This function is deprecated since AUTOSAR 4.3.0. Use 'Com_ReceiveSignal' instead to read group signals.	
Call context	
> The function can be called on interrupt and task level. > This function is Synchronous	

Table 4-28 Com_ReceiveShadowSignal

4.2.28 Com_UpdateShadowSignal

Prototype	
`uint8 **Com_UpdateShadowSignal** (Com_SignalIdType SignalId, const void *SignalDataPtr)`	
Parameter	
SignalId [in]	ID of group signal to be updated.
SignalDataPtr [in]	Reference to the group signal data to be updated.
Return code	
uint8	void
Functional Description	
The service Com_UpdateShadowSignal updates a group signal with the data, referenced by SignalDataPtr. The update of the group 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.	
Particularities and Limitations	
This function is deprecated since AUTOSAR 4.3.0. Use 'Com_SendSignal' instead to send group signals.	
Call context	
> The function can be called on interrupt and task level. > This function is Synchronous	

Table 4-29 Com_UpdateShadowSignal

4.2.29 Com_InvalidateShadowSignal

Prototype	
`uint8 **Com_InvalidateShadowSignal** (Com_SignalIdType SignalId)`	
Parameter	
SignalId [in]	ID of group signal to be invalidated.
Return code	
uint8	void
Functional Description	
This function invalidates the group signal by calling Com_SendSignal with the configured invalid value. An additional call to Com_SendSignalGroup is needed to update the signal group data.	
Particularities and Limitations	
This function is deprecated since AUTOSAR 4.3.0. Use 'Com_InvalidateSignal' instead to invalidate group signals.	
Call context	
> The function can be called on interrupt and task level and has not to be interrupted by other Com_SendSignal and Com_InvalidateSignal calls for the same SignalId. > This function is Synchronous	

Table 4-30 Com_InvalidateShadowSignal

4.3 Services used by COM

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

Component	API
PDUR	PduR_ComTransmit
DET	Det_ReportError
Application	I-PDU Callout
RTE/Application	Signal and ComSignalGroup configurable callback/notification functions
EcuM	EcuM_BswErrorHook

Table 4-31 Services used by the COM

4.4 Callback Functions

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

4.4.1 Com_RxIndication

Prototype	
`void` **`Com_RxIndication`** `(PduIdType RxPduId, const PduInfoType* PduInfoPtr)`	
Parameter	
RxPduId [in]	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
PduInfoPtr [in]	PduInfoPtr Payload information of the received I-PDU (pointer to data and data length).
Return code	
void	none
Functional Description	
This function is called by the lower layer after an I-PDU has been received.	
Particularities and Limitations	
The function is called by the lower layer.	
Call Context	
The function can be called on interrupt and task level. It is not allowed to use CAT1 interrupts with Rte (BSW00326]). Due to this, the interrupt context shall be configured to a CAT2 interrupt if an Rte is used.	

Table 4-32 Com_RxIndication

4.4.2 Com_TxConfirmation

Prototype	
`void **Com_TxConfirmation** (PduIdType TxPduId)`	
Parameter	
TxPduId [in]	ID of AUTOSAR COM I-PDU that has been transmitted. Range: 0..(maximum number of I-PDU IDs transmitted by AUTOSAR COM) - 1
Return code	
void	none
Functional 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.	
Particularities and Limitations	
The function is called by the lower layer.	
Call Context	
The function can be called on interrupt and task level. It is not allowed to use CAT1 interrupts with Rte (BSW00326]). Due to this, the interrupt context shall be configured to a CAT2 interrupt if an Rte is used.	

Table 4-33 Com_TxConfirmation

4.4.3 Com_TriggerTransmit

Prototype	
`Std_ReturnType` **`Com_TriggerTransmit`** `(PduIdType TxPduId, PduInfoType *PduInfoPtr)`	
Parameter	
TxPduId [in]	ID of AUTOSAR COM I-PDU that is requested to be transmitted by AUTOSAR COM.
PduInfoPtr [in, out]	Contains a pointer to a buffer (SduDataPtr) where the SDU data shall be copied to, and the available buffer size in SduLengh. On return, the service will indicate the length of the copied SDU data in SduLength.
Return code	
Std_ReturnType	E_OK: SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK: No data has been copied, because Com is not initialized or TxPduId is not valid or PduInfoPtr is NULL_PTR or SduDataPtr is NULL_PTR or SduLength is too small.
Functional 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. 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().	
Particularities and Limitations	
The function is called by the lower layer.	
Call Context	
The function can be called on interrupt and task level. It is not allowed to use CAT1 interrupts with Rte (BSW00326]). Due to this, the interrupt context shall be configured to a CAT2 interrupt if an Rte is used.	

Table 4-34 Com_TriggerTransmit

4.4.4 Com_TpTxConfirmation

Prototype	
`void` **`Com_TpTxConfirmation`** `(PduIdType PduId, Std_ReturnType Result)`	
Parameter	
PduId [in]	ID of the I-PDU that has been transmitted.
Result [in]	Result of the transmission of the I-PDU
Return Code	
void	None.
Functional Description	
This function is called by the PduR after a large I-PDU has been transmitted via the transport protocol on its network.	
Particularities and Limitations	
The function is called by the lower layer.	
Pre-Conditions	
Call Context	
The function can be called on interrupt and task level.	

Table 4-35 Com_TpTxConfirmation

4.4.5 Com_CopyTxData

Prototype	
`BufReq_ReturnType` **`Com_CopyTxData`** `(PduIdType PduId, PduInfoType *PduInfoPtr, RetryInfoType *RetryInfoPtr, PduLengthType *TxDataCntPtr)`	
Parameter	
PduId [in]	ID of Com TP I-PDU to be transmitted.
PduInfoPtr [in]	Pointer to a PduInfoType, which indicates the number of bytes to be copied (SduLength) and the location where the data have to be copied to (SduDataPtr). An SduLength of 0 is possible in order to poll the available transmit data count. In this case no data are to be copied and SduDataPtr might be invalid.
RetryInfoPtr [in]	If the TpDataState of the RetryInfoPtr is TP_DATARETRY, no data will be copied. Otherwise, the COM module will ignore the value of this pointer, since it always keeps the complete buffer until the transmission of a large I-PDU is either confirmed or aborted.
TxDataCntPtr [out]	Out parameter: Remaining Tx data after completion of this call.
Return Code	
BufReq_ReturnType	BufReq_ReturnType BUFREQ_OK: Data has been copied to the transmit buffer completely as requested. BUFREQ_E_BUSY: The transmission buffer is actually not available (implementation specific). BUFREQ_E_NOT_OK: Data has not been copied. Request failed, in case the corresponding I-PDU was stopped.
Functional Description	
This function is called by the lower layer to copy Data from the Com TP buffer to the lower layer TP buffer.	
Particularities and Limitations	
The function is called by the lower layer.	
Pre-Conditions	
Call Context	
The function can be called on interrupt and task level.	

Table 4-36 Com_CopyTxData

4.4.6 Com_TpRxIndication

Prototype	
`void Com_TpRxIndication (PduIdType PduId, Std_ReturnType Result)`	
Parameter	
PduId [in]	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
Result [in]	Indicates whether the Message was received successfully.
Return Code	
void	none
Functional Description	
This function is called by the lower layer after a TP I-PDU has been received.	
Particularities and Limitations	
The function is called by the lower layer.	
Pre-Conditions	
Call Context	
The function can be called on interrupt and task level. It is not allowed to use CAT1 interrupts with Rte (BSW00326]). Due to this, the interrupt context shall be configured to a CAT2 interrupt if an Rte is used.	

Table 4-37 Com_TpRxIndication

4.4.7 Com_StartOfReception

Prototype	
`BufReq_ReturnType` **`Com_StartOfReception`** `(PduIdType ComRxPduId, PduInfoType* TpSduInfoPtr, PduLengthType TpSduLength, PduLengthType *RxBufferSizePtr)`	
Parameter	
ComRxPduId [in]	ID of AUTOSAR COM I-PDU that has been received. Identifies the data that has been received.
TpSduInfoPtr [in]	Is currently not used by COM.
TpSduLength [in]	complete length of the TP I-PDU to be received.
RxBufferSizePtr [out]	The Com returns in this value the remaining TP buffer size to the lower layer.
Return Code	
BufReq_ReturnType	BufReq_ReturnType BUFREQ_OK: Connection has been accepted. RxBufferSizePtr indicates the available receive buffer. BUFREQ_E_NOT_OK: Connection has been rejected. RxBufferSizePtr remains unchanged. BUFREQ_E_OVFL: In case the configured buffer size as specified via ComPduIdRef.PduLength is smaller than TpSduLength. BUFREQ_E_BUSY: In case the reception buffer is actually not available for a new reception (implementation specific).
Functional Description	
This function is called by the lower layer to indicate the start of a incomming TP connection.	
Particularities and Limitations	
The function is called by the lower layer.	
Pre-Conditions	
Call Context	
The function can be called on interrupt and task level.	

Table 4-38 Com_StartOfReception

4.4.8 Com_CopyRxData

Prototype	
`BufReq_ReturnType` **`Com_CopyRxData`** `(PduIdType PduId, const PduInfoType *PduInfoPointer, PduLengthType *RxBufferSizePtr)`	
Parameter	
PduId [in]	ID of AUTOSAR COM I-PDU that has been received. Identifies the data that has been received.
PduInfoPointer [in]	Payload information of the received TP segment (pointer to data and data length).
RxBufferSizePtr [out]	The Com returns in this value the remaining TP buffer size to the lower layer.
Return Code	
BufReq_ReturnType	BufReq_ReturnType BUFREQ_OK: Connection has been accepted. RxBufferSizePtr indicates the available receive buffer. BUFREQ_E_NOT_OK: Connection has been rejected. RxBufferSizePtr remains unchanged.
Functional Description	
This function is called by the lower layer to hand a received TP segment to Com. The Com copies the received segment in his internal tp buffer.	
Particularities and Limitations	
The function is called by the lower layer.	
Pre-Conditions	
Call Context	
The function can be called on interrupt and task level.	

Table 4-39 Com_CopyRxData

4.5 Configurable Interfaces

4.5.1 Notifications

At its configurable interfaces the COM defines notifications that can be mapped to callback functions provided by other modules. The mapping is not statically defined by the COM but can be performed at configuration time. The function prototypes that can be used for the configuration have to match the appropriate function prototype signatures, which are described in the following sub-chapters.

4.5.1.1 Indication Notification

Prototype	
`void` **`[Indication Notification Name]`** `( void )`	
Parameter	
void	none
Return code	
void	none

Functional Description
The notification function is called after the message has been received successfully. The function can be configured for signals and signal groups with a configurable function name.
Particularities and Limitations
> none
Call Context
> The call context depends on the configuration of the parameter ComIPduSignalProcessing for the I-PDU.

Table 4-40 Indication Notification

4.5.1.2 Confirmation Notification

Prototype	
`void` **`[Confirmation Notification Name]`** `( void )`	
Parameter	
void	none
Return code	
void	none
Functional Description	
The notification function is called after successful transmission of the I-PDU containing the message. The function can be configured for signals and signal groups with a configurable function name.	
Particularities and Limitations	
> none	
Call Context	
> The call context depends on the configuration of the parameter ComIPduSignalProcessing for the I-PDU.	

Table 4-41 Confirmation Notification

4.5.1.3 Rx Timeout Notification

Prototype	
`void` **`[Rx Timeout Notification Name]`** `( void )`	
Parameter	
void	none
Return code	
void	none
Functional Description	
It is called immediately after a message reception error has been detected by the deadline monitoring mechanism. The function can be configured for signals with a configurable function name.	

Particularities and Limitations
> none
Call Context
> The function is called on task level.

Table 4-42 Rx Timeout Notification

4.5.1.4 Tx Timeout Notification

Prototype	
`void [Tx Timeout Notification Name] ( void )`	
Parameter	
void	none
Return code	
void	none
Functional Description	
It is called immediately after a message transmission error has been detected by the deadline monitoring mechanism. The function can be configured for signals and signal groups with a configurable function name.	
Particularities and Limitations	
> none	
Call Context	
> The function is called on task level.	

Table 4-43 Tx Timeout Notification

4.5.1.5 Error Notification

Prototype	
`void [Error Notification Name] ( void )`	
Parameter	
void	none
Return code	
void	none
Functional Description	
It is called immediately for outstanding, not confirmed transmitted signals that are contained in I-PDUs that get stopped by a call to Com_IpduGroupStop.	
Particularities and Limitations	
> none	
Call Context	
> The function is in the context of Com_IpduGroupStop.	

Table 4-44 Error Notification

4.5.1.6 Invalid Notification

Prototype	
`void [Invalid Notification Name] ( void )`	
Parameter	
void	none
Return code	
void	none
Functional Description	
This notification is called, if an invalid value is received and the invalid action is configured to 'NOTIFY'.	
Particularities and Limitations	
> none	
Call Context	
> The call context depends on the configuration of the parameter ComIPduSignalProcessing for the I-PDU.	

Table 4-45 Invalid Notification

4.5.2 Callout Functions

At its configurable interfaces the COM defines callout functions. The declarations of the callout functions are provided by the BSW module, i.e. the COM. It is the integrator's task to provide the corresponding function definitions. The definitions of the callouts can be adjusted to the system's needs. The COM callout function declarations are described in the following tables:

4.5.2.1 Rx I-Pdu callout function

The Rx I-PDU callouts are implemented as specified by AUTOSAR 4.1.1

Prototype	
`boolean [IPDU_CalloutName] (PduIdType PduId, const PduInfoType* PduInfoPtr)`	
Parameter	
PduId [in]	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
PduInfoPtr [in]	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.
Return code	
boolean	The return value indicates whether the processing of this I-PDU shall continue (TRUE) or abandon (FALSE) after the callout returns.
Functional Description	
For each I-PDU a callout function can be configured and the implementation has to be provided by the application. It can be used to extend the COM functionality with application related functions (e.g. CRC or sequence counter calculation).	

Particularities and Limitations
> In this context, the signal access APIs could not be used to read the signal values, as the internal buffer is not updated yet and the old values are returned.
Call Context
> Rx I-PDU Callouts are called by COM directly after Com_RxIndication on task or interrupt level. The call context depends on the configuration of the Driver.

Table 4-46 Rx I-PDU Callout with PduInfo pointer

4.5.2.2 Tx I-Pdu callout function

The Tx I-PDU callouts are implemented as specified by AUTOSAR 4.1.1

Prototype	
`boolean` **`[IPDU_CalloutName]`** `(PduIdType PduId, PduInfoType* PduInfoPtr)`	
Parameter	
PduId [in]	ID of AUTOSAR COM I-PDU that should be transmitted. Identifies the data that should be transmitted. Range: 0..(maximum number of I-PDU IDs transmitted by AUTOSAR COM) - 1
PduInfoPtr [in, out]	Contains the length (SduLength) of the I-PDU to be transmitted and a pointer to a buffer (SduDataPtr) containing the I-PDU.
Return code	
boolean	The return value indicates whether the processing of this I-PDU shall continue (TRUE) or abandon (FALSE) after the callout returns.
Functional Description	
For each I-PDU a callout function can be configured and the implementation has to be provided by the application. It can be used to extend the COM functionality with application related functions (e.g. CRC or sequence counter calculation).	
Particularities and Limitations	
> In this context the signal access APIs can be used to update the signal values, as these APIs directly updates the internal buffer. If a triggered event is caused by such a call, no additional transmission of the I-PDU is triggered.	
Call Context	
> Tx I-PDU Callouts are called by COM directly before the call of PduR_ComTransmit on task level. > Tx I-PDU Trigger Transmit Callouts are called by COM in the function Com_TriggerTransmit. The call context depends on the configuration of the lower layer.	

Table 4-47 Tx I-PDU Callout with PduInfo pointer

5 Configuration

5.1 Configuration Variants

The COM supports the configuration variants

- `VARIANT-PRE-COMPILE`
- `VARIANT-LINK-TIME`
- `VARIANT-POST-BUILD-LOADABLE`
- `VARIANT-POST-BUILD-SELECTABLE`

The configuration classes of the COM parameters depend on the supported configuration variants. For their definitions please see the Com_bswmd.arxml file.

5.2 Configuration of Post-Build

The configuration of post-build loadable is described in [5].

5.3 Signal Routing to Description Routing

The VASE script is used to derive description routing from signal routing. In order to use the script a description routing license is required.

6 Migration

This chapter describes the tasks to be done for a migration of the Com from a version prior 22.00.00 to the version 22.00.00 or later.

With the version 22.00.00 the configuration of multiple MainFunctions is possible. Because of this change, the names of the MainFunctions change and the MainFunction to Task Mapping is lost. If you want to preserve the MainFunction to Task Mapping you must do the following steps for the migration of your configuration.

- Do the migration of your configuration.
- Run the script task RestoreReferencesRTEEVentToTaskMappings before you update the BswInternalBehavior of the Com.

7 Glossary and Abbreviations

7.1 Glossary

Term	Description
CFG5	DaVinci Configurator

Table 7-1 Glossary

7.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
EAD	Embedded Architecture Designer
ECU	Electronic Control Unit
HIS	Hersteller Initiative Software
ISR	Interrupt Service Routine
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
RTE	Runtime Environment
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification

Table 7-2 Abbreviations

8 Contact

Visit our website for more information on

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

www.vector.com