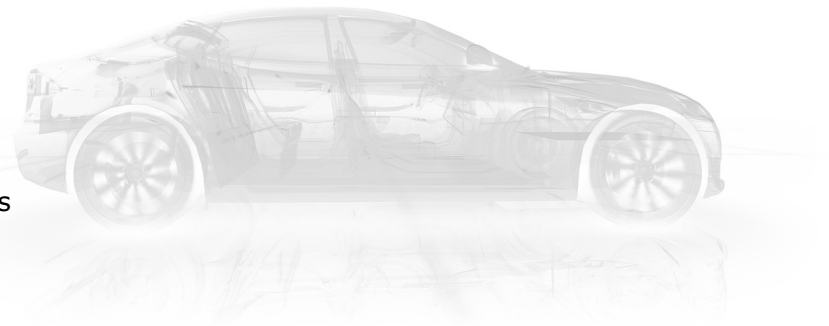
NIT

AUTOMOTIVE SOFTWARE WITH AUTOSAR

AUTOSAR

Agenda

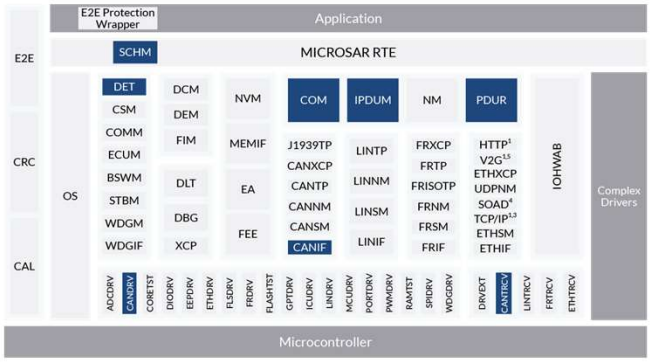
- **Communication**
- Transmission
- Reception
- Signal Groups
- Update Bit
- Data Mapping
- Notification Mechanisms
- Deadline Monitoring
- Invalidation
- Additional Return Values



Communication

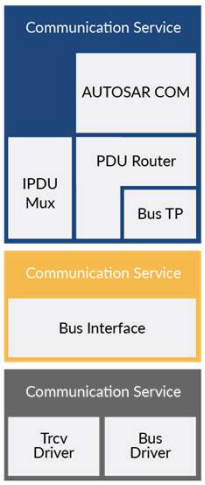
Related modules

- COM
- IPDUM
- PDUR
- CANIF
- CAN
- CANTRCV
- DET
- SCHM



COM — Communication
PDUR — PDU Router
CANIF — CAN Interface
CAN — CAN Driver
DET — Development Error Tracer
SCHM — Schedule Manager

Communication
Communication modules interaction



Com operates on signals but has no awareness of the underlying bus technology. Com module implements Com_SendSignal and Com_ReceiveSignal, which will trigger send of message (PDU) over bus (signals are fragments of message).

PDU Router is responsible for routing of messages (for same or different type of buses, i.e., CAN -> CAN, CAN -> FlexRay) and between service layer modules (Com/Dcm) and transport protocol modules.
PDU router abstracts bus system technology for higher layers.

Transport protocol is used for segmentation and desegmentation. It is only used by the Diagnostic Communication Manager (Dcm) to allow bigger messages then allowed by underlying bus technology.

Maximum size of message is limited by the underlying communication interface:
CAN - 8B
LIN – 8B
CANFD – 64B
FlexRay – 254B

Driver consists of a hardware-specific driver (i.e., CAN-DRV) and the hardware-independent interface (i.e., CAN-IF).
Hardware-independent driver is implementing additional queuing, microcontroller state control handling and generic APIs to hardware-specific driver.
Hardware-specific driver is implementing microcontroller initialization, access to buffers and ISR routines.

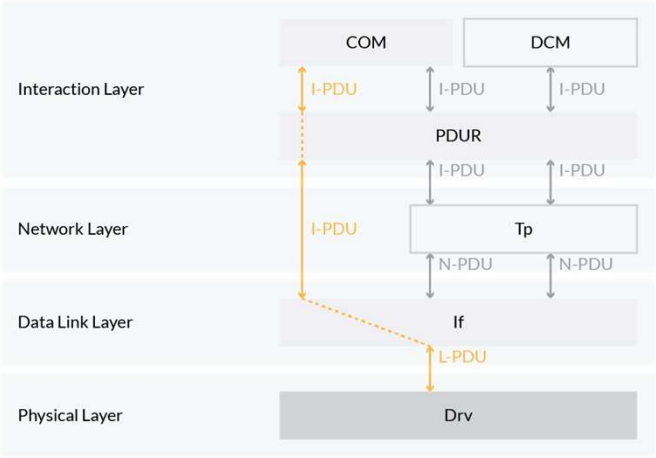
CAN Transceiver driver adapts the signals on the CAN bus to the digital signals recognized by the microcontroller.
In addition, the transceivers can detect electrical malfunctions like wiring issues.

DET collects all detected errors of the BSW modules, used during development only.

SchM is BSW module that triggers main processing functions of the BSW modules.

IPDUM handles messages with multiple different signal layouts.

Communication
PDU concept



I-PDU is Interaction Layer Protocol Data Unit, it consists of data, length and ID.
The PDU router will mainly route I-PDUs.
I-PDUs are used for the data exchange for Com and Dcm (Diagnostic Communication Manager).

N-PDU is Network Layer Protocol Data Unit. It is used in the AUTOSAR Transport Protocol Layer to fragment I-PDU.
Frames that are used for fragmentation of I-PDU:

- SF - single frame
- FF - first frame
- CF - consecutive frame

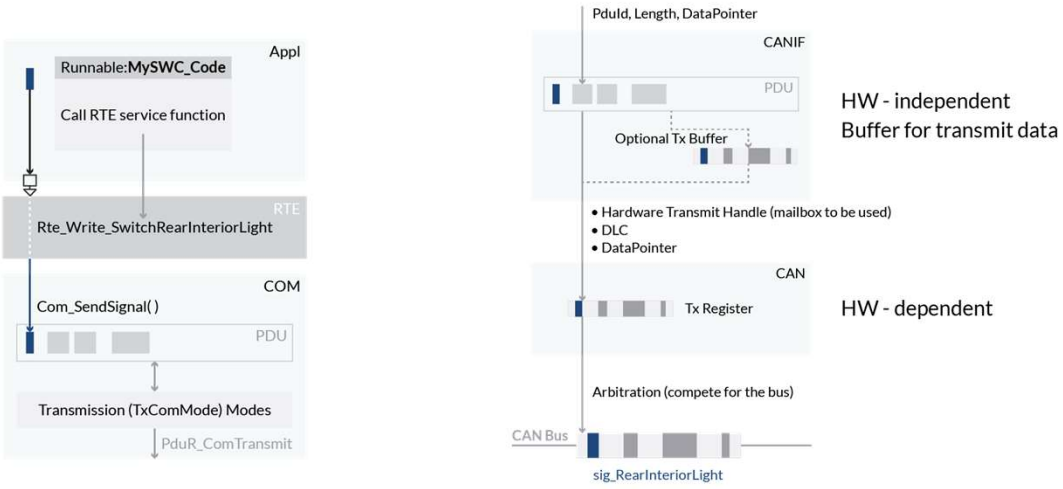
L-PDU is a Data Link Layer Protocol Data Unit which is assembled and disassembled in Hardware Abstraction layer (bus driver). One or more I-PDUs are packed into one L-PDU.
The L-PDU is bus specific, e.g., CAN frame.

For CAN bus L-PDU and I-PDU are usually the same.

Agenda

- Communication
- **Transmission**
- Reception
- Signal Groups
- Update Bit
- Data Mapping
- Notification Mechanisms
- Deadline Monitoring
- Invalidation
- Additional Return Values

Transmission
Signal Transmission



Sender side will call `Rte_Write_<SWC>_<Port>_<DataElement>` from which `Com_SendSignal()` is called. `Com_SendSignal()` will write data value into Com buffer and indicate to lower layers that new signal is available.

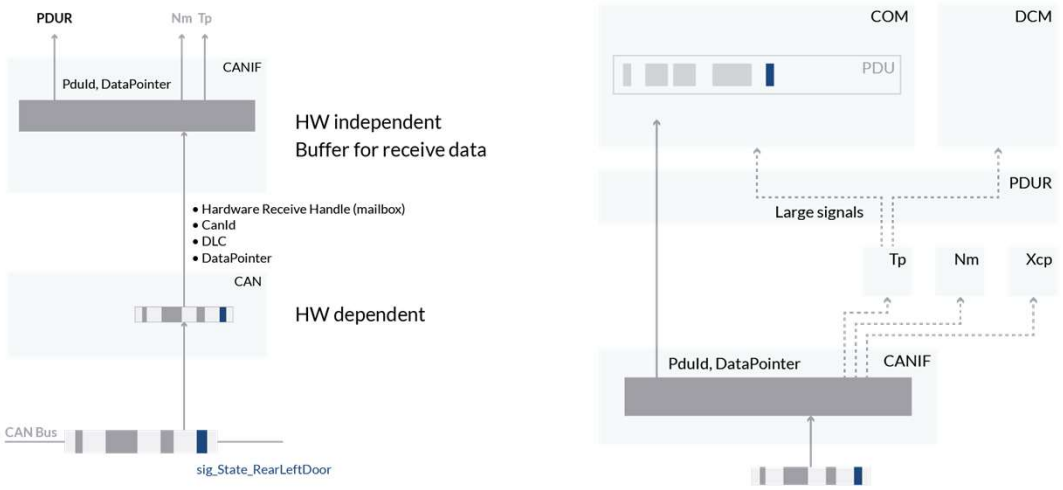
Signal/PDU will be sent over bus with interaction of following functions (respectively):

- `Com_MainFunctionTx`
- `PduR_ComTransmit`
- `CanIf_Transmit`
- `Can_Write`

Agenda

- Communication
- Transmission
- **Reception**
- Signal Groups
- Update Bit
- Data Mapping
- Notification Mechanisms
- Deadline Monitoring
- Invalidation
- Additional Return Values

Reception
Signal Reception



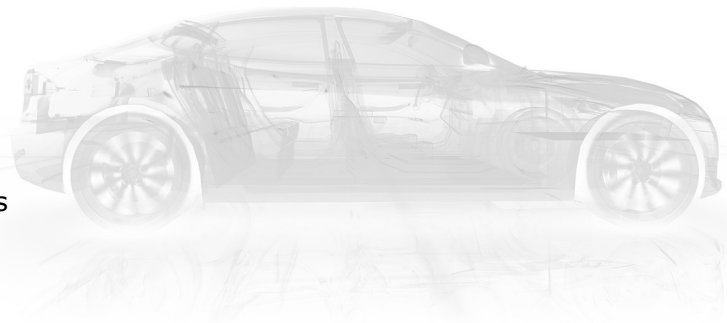
On receiver side `Rte_Read_<SWC>_<Port>_<DataElement>` will call `Com_ReceiveSignal()`. `Com_ReceiveSignal()` will read value from Com buffer and return it to caller (`Rte_Read_<SWC>_<Port>_<DataElement>`).

Signal/PDU will be received over bus with the following functions (respectively):

- `CanInterruptRx`
- `CanIf_RxIndication`
- `PduR_CanIfRxIndication`
- `Com_RxIndication`

Agenda

- Communication
- Transmission
- Reception
- **Signal Groups**
- Update Bit
- Data Mapping
- Notification Mechanisms
- Deadline Monitoring
- Invalidation
- Additional Return Values



Signal Groups

Signal groups

› Logical group of signals that belong together
› Signal groups are defined in the communication matrix
› Complex data types are mapped to signal groups by the RTE

Signal Groups signify the following

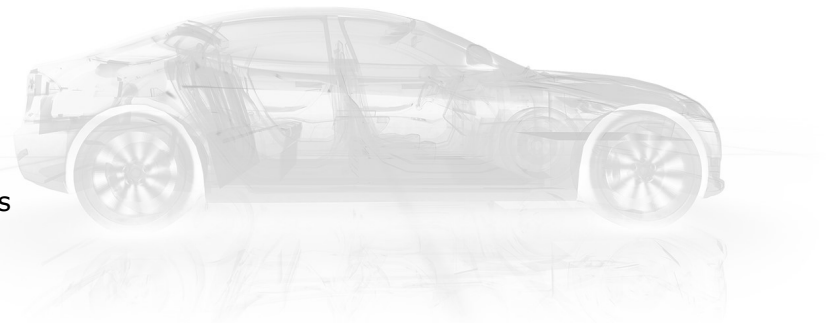
› All group signals within a signal group are consistent
› Access is done by the RTE via a dedicated signal group and group signal API
› Signal group shadow buffer provided by Com, one global shadow buffer for each signal group

API `Com_SendSignal()` writes a group signal into the shadow buffer, not directly to the I-PDU.

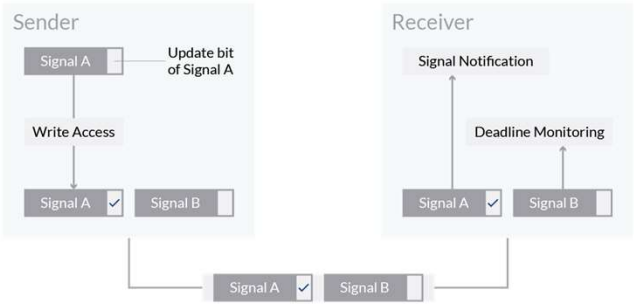
After all group signals of a signal group have been updated, the consistent content of the shadow buffer is "transmitted" using the API `Com_SendSignalGroup()`. A similar API is available for signal group reception.

Agenda

- Communication
- Transmission
- Reception
- Signal Groups
- **Update Bit**
- Data Mapping
- Notification Mechanisms
- Deadline Monitoring
- Invalidation
- Additional Return Values



Update Bit
Concept

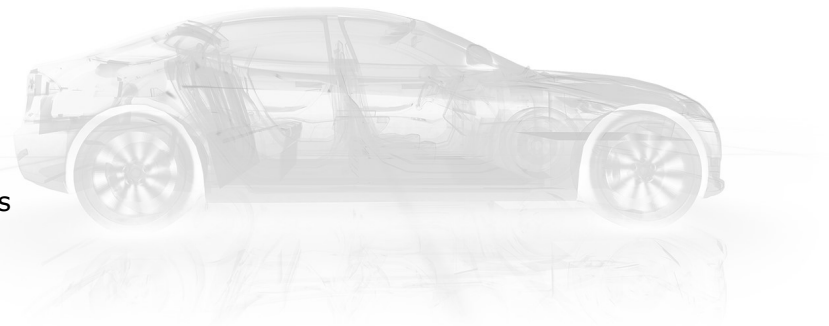


Update Bit is defined within the communication matrix as dedicated bit within the I-PDU of the related signal. It allows the receiver to determine if the transmitter has updated the signal value.

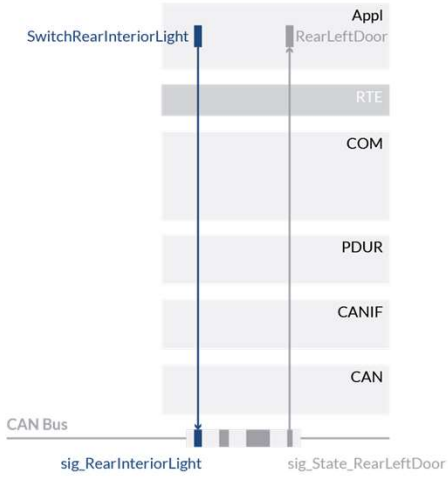
Update bit can't be directly accessed from the application or RTE, it is set by Com when signal is written, and cleared when PDU is successfully transmitted.
Usage of update bit allows notification of callback functions and deadline monitoring (detect if signal is obsolete and execute correct actions).

Agenda

- Communication
- Transmission
- Reception
- Signal Groups
- Update Bit
- **Data Mapping**
- Notification Mechanisms
- Deadline Monitoring
- Invalidation
- Additional Return Values



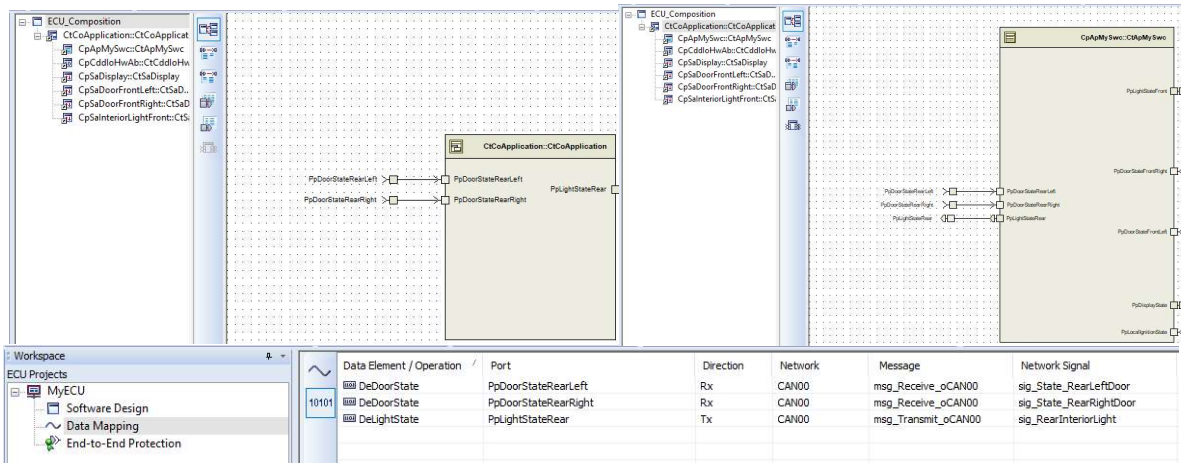
Data Mapping
Overview



Data Elements sent over bus have to be mapped to Network Signals.
Process of assigning Data Elements inside Delegation Ports (of ECU_Composition) to bus signal is called Data Mapping.

Delegation ports are extensions of regular ports that are exiting current composition (providing/receiving data outside composition).

Data Mapping
Data mapping in DaVinci Developer



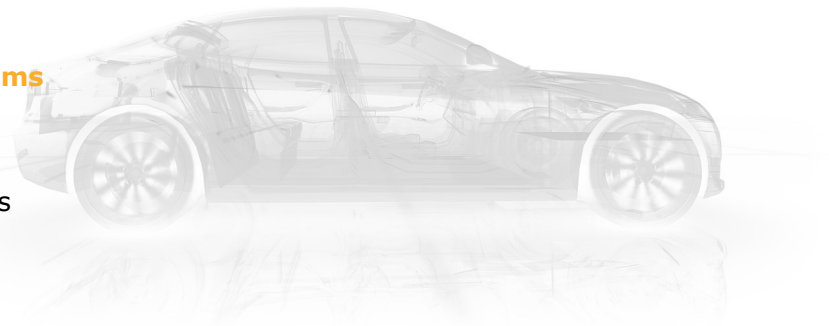
If SWC, that is sending/receiving signals, is inside additional composition additional delegation ports need to be created.

Delegation ports are used to provide/receive data outside current composition
(one delegation port to exit CtCoApplication and one delegation port to exit ECU_Compostion).

Data elements can be mapped to signals only if they are part of Delegation Ports at the ECU Composition level.

Agenda

- Communication
- Transmission
- Reception
- Signal Groups
- Update Bit
- Data Mapping
- **Notification Mechanisms**
- Deadline Monitoring
- Invalidation
- Additional Return Values



Notification Mechanisms

I-PDU Signal Processing

I-PDU Signal Processing determines the call context and the point in time when the signal's Rx notification or Tx notification is called.

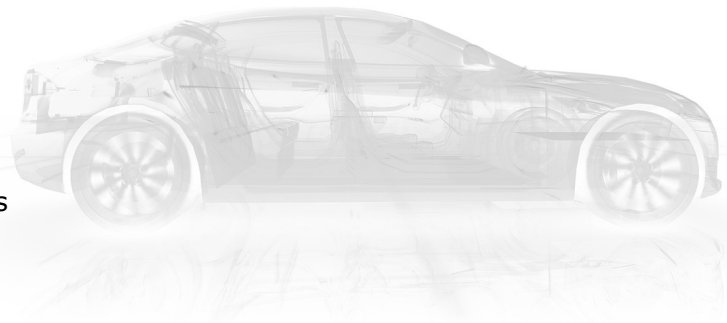
There are two types I-PDU Signal Processing:

› IMMEDIATE, notification function is called within `Com_RxIndication()` or `Com_TxConfirmation()`, from interrupt context

› DEFERRED, notification function is called on task level during the next call cycle of `Com_MainFunctionRx()` or `Com_MainFunctionTx()`

Agenda

- Communication
- Transmission
- Reception
- Signal Groups
- Update Bit
- Data Mapping
- Notification Mechanisms
- **Deadline Monitoring**
- Invalidation
- Additional Return Values



Deadline Monitoring
Mechanism and adjustment of time value

COM can detect timeouts on:

› I-PDU level, Rx I-PDU is not received within a predefined timeout time

› signal level, update bit of a signal has not been set for a predefined timeout time

Timeout time:

› can be configured in DaVinci Developer and Configurator

› is derived from the smallest timeout of monitored signals within the I-PDU

Deadline Monitoring
Timeout on COM and RTE level

COM Timeout actions:

› timeout callback function is called by Com, `Rte_COMCbkTOut_<ComSignalName>`

› affected signals are set to their initial values

RTE Timeout handling:

› timeout can trigger a runnable in the RTE

› return value of the function `Rte_Read_<Port>_<DataElement>()` indicates the error `RTE_E_MAX_AGE_EXCEEDED`

For intra-ECU Sender/Receiver communication, the RTE does not perform timeout monitoring.

RTE can do timeout supervision for asynchronous, intra-ECU Client/Server API calls.

If Com detects a timeout, it will not set any Dem event automatically. If a Com timeout shall result in a Dem entry, the application has to set the Dem event by itself.

Com calls the timeout notification cyclically if the timeout condition exists. The cycle time is equal to the configured timeout time.

Agenda

- Communication
- Transmission
- Reception
- Signal Groups
- Update Bit
- Data Mapping
- Notification Mechanisms
- Deadline Monitoring
- **Invalidation**
- Additional Return Values

Invalidation
RTE level

Invalidation can be used only for non-queued communication

Inter-ECU and Intra-ECU supported

Send invalid data via
- `Rte_Invalidate` (explicit API)
- `Rte_IInvalidate` (implicit API)

Handle Invalid receive data
- KEEP
- REPLACE
- NONE

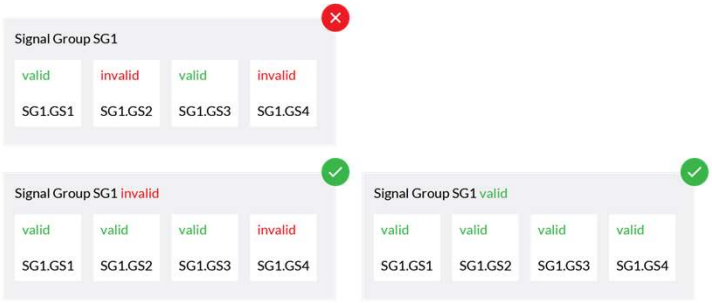
KEEP - `RTE_E_INVALID` returned, and last valid value kept in COM

REPLACE – replaces invalid value with Init value in COM

NONE - `RTE_E_INVALID` not returned, invalid COM value propagates to application

Invalidation
COM level

Signal group is **invalid** if only one signal inside group is invalid.

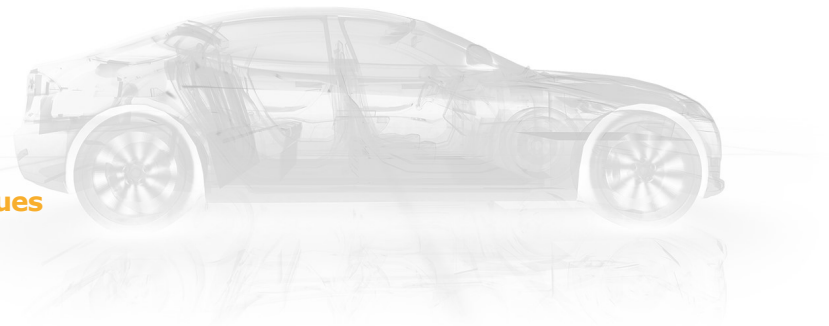


Invalid actions on Com level

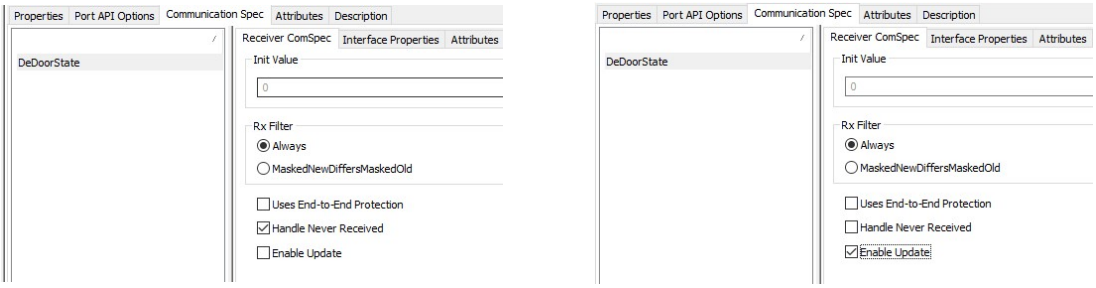
- NONE - nothing happens received invalid value is in the Com buffer
- NOTIFY - the invalid notification `Rte_COMCbkInv_<ComSignalName>` will be called. Last valid received value remains in the Com buffer
- REPLACE - replace the received invalid value in the Com buffer by the Init value

Agenda

- Communication
- Transmission
- Reception
- Signal Groups
- Update Bit
- Data Mapping
- Notification Mechanisms
- Deadline Monitoring
- Invalidation
- **Additional Return Values**



Additional Return Values
Further Communication Specification settings



In DaVinci Developer for each data elements of receiver port go to Communication Spec where you can find multiple flags that can add additional communication interfaces.

Check “Handle Never Received“ checkbox to get `Rte_IStatus` API.
`Rte_IStatus` will return `RTE_E_NEVER_RECEIVED` if no data received over COM, since `Rte_Start`.
`Rte_IRead` provides the Init value in the case of `RTE_E_NEVER_RECEIVED`.
`Rte_IStatus` comes in pair with Rte_IRead (implicit access).

Check "Enable Update" checkbox to get `Rte_IsUpdated`.
`Rte_IsUpdated` tells if new data is received.
`Rte_IsUpdated` comes in pair with Rte_Read (direct access).