Document Title	Explanation of Application Interfaces of the Body and Comfort Domain
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Identification No	268
Document Classification	Auxiliary

Document Version	1.0.0
Document Status	Final
Part of Release	3.0
Revision	0001

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
04.12.2007	1.0.0	AUTOSAR Administration	Initial Release

Page left intentionally blank

Disclaimer

Advice to users of AUTOSAR Specification Documents:

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).
Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later AUTOSAR compliance certification of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Purpose of this Document

This document explains all design decisions that lead to the MasterTable contents.

In case of inconsistencies between pictures and explanations in this document and the MasterTable, the information in the MasterTable has to be considered as prior.

2 Description of Terms and Concepts of the Body- and Comfort Domain

2.1 Design Rationale

This design standard is the AUTOSAR-architecture (RTE-view) for Software Compositions described in chapter 4. It provides a decomposition into SW-Cs and a list of standardized interfaces related to each functionality (like Interior Lights).

This decomposition is limited in granularity to sensor components, adapter components, the core functionality and actuator components.

The intent is for the decomposition to get not to atomic SWCs but to “purchasable” SW-Cs (as long as these are not sensor/actuator SW-Cs they are realized as AUTOSAR compositions for formal reasons). These SW-Cs will be obtained as a unit so that all the internal (and hence not AUTOSAR standardized) interfaces are controlled by a single vendor, even if there are SW-Cs within the bought unit that reside on different ECUs. All interfaces between SW-Cs from different vendors should be standardized.

In addition, all ports are described, showing their AUTOSAR data qualities. Invalidation (where needed) is defined as in-band invalidation. Init values are specified where appropriate e.g. “off”, “idle”, “undefined”, “unknown” … Where standardization is inappropriate, recommendations may be provided.

2.2 Naming Convention[1]

Guidelines for the naming convention:

- Type safety, i.e. interface definitions reflect usage/classification. Implausible connections are more difficult to make unintentionally.
- Easy finding of names (e.g. data element name is just one of „status“, „command“, „request“, „display“)
- Keep names simple whenever possible. Extend names only if required (e.g. multiple data elements).

	Port	***Interface***	***Data element***
Single occurrence	Content	Operation + Content	Operation
Example	`LowBeamInd` `ParkingLightsInd`	`DisExtLight` `DisParkingLights`	`display` `display`

	Port	***Interface***	***Data element***
Multiple occur-	Content + (ExtendedCon-	Operation + Content	Operation + Content

[1] This chapter Naming Convention may no longer exist after R3.0 and may be replaced by the AUTOSAR SW-C & System Modeling Guide in the upcoming AUTOSAR Release

rence	tent)		
Example for multiple ports	`[IntLight] Activa-tion`	`CmdBrightness`	`command`

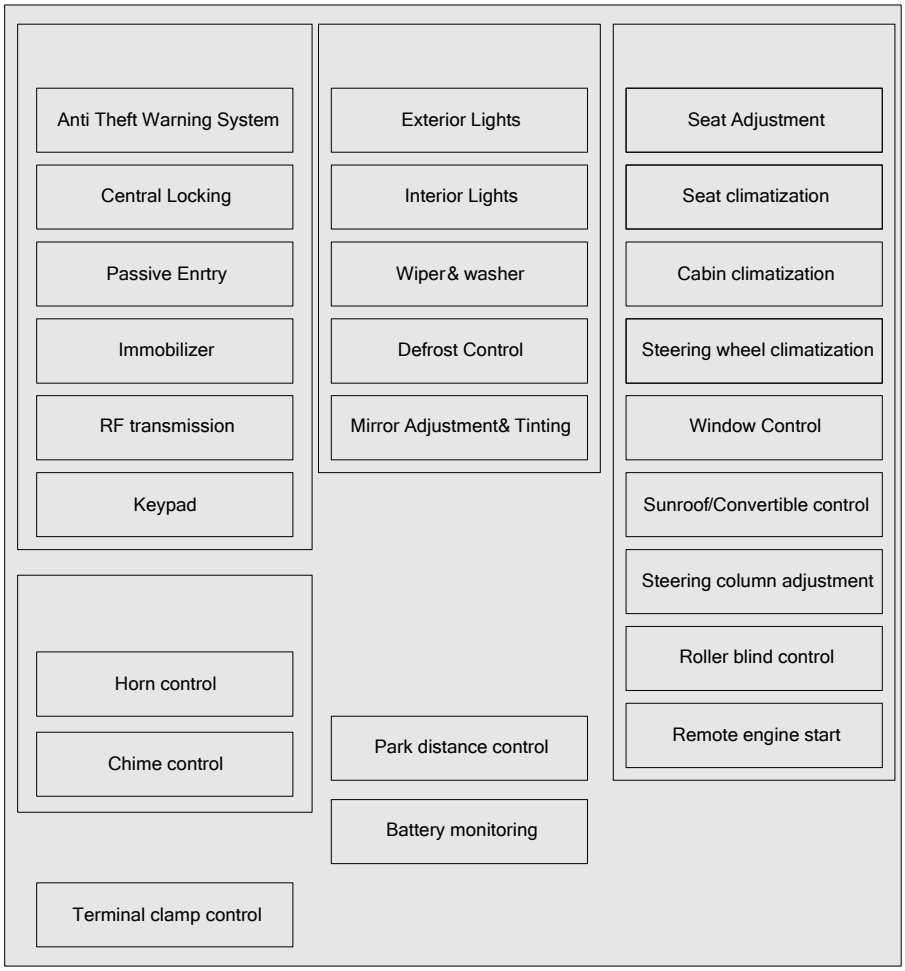
Semantic of keywords (e.g. “operation”) in the interface/ data element names:

- **Cmd**(command) do/activate something (e. g. from Master to Actuator)
- **Req**(request) demand to do/activate something (e. g. from Sensor to Master)
- **Sta**(status) get functional status information
- **Hmi** user request (e.g. from driver via switch, touch screen,…)
- **Dis**(display) feedback status for driver information display
- **Err**(failure) operative/defective failure feedback (from actuator to master)

The “content” consists of one or more self-explaining catchwords.

3 Architecture Overview

This section gives an overview of the top level de-composition of the Body and Comfort Domain.



The function components from Body and Comfort domain are identified according to the controlled group of actuators (motors, bulbs and other special devices) and type of operations (car access, lighting, comfort, acustic signalling and other indirect operations like Terminal clamp control or battery monitoring). These function components can be grouped also in different ways considering other criterias.

Acce
subsys

The defined subsystems does not always mean a strong link between the included function components. The subsystems are also linked between them by a set of interfaces. For example the Access subsystem is strongly linked with a part of the Visibility subsystem.

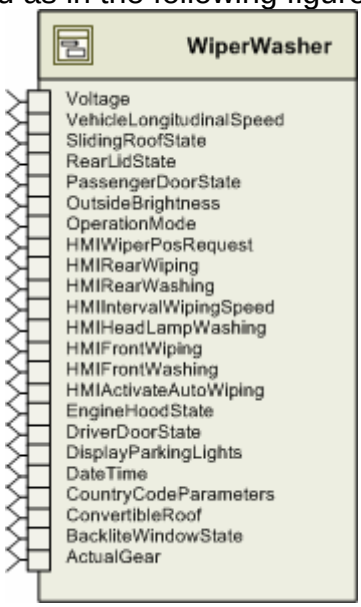
There is a considerable complexity of interconnections inside body domain but also a set of interfaces are provided and required from other domains.

4 Description of Software Compositions and Components

The following Software Compositions and Components were identified in the body- and comfort domain.

4.1 Wiper/Washer

The `WiperWasher` software component group controls the wiper and washer functionality of a car. `WiperWasher` receives driver wishes, senses the environment and controls the behaviour of the wipers and washers of a car. The top-level component can be graphically represented as in the following figure.



Components related to `WiperWasher` are described in the following sections.

`ExteriorLight` provides information about the outside brightness for the `Rain Sensing` component.
`ClampControl` and `BatteryMonitor` provide information about the current state of the vehicle.
`Rain Sensing`, consists of the Rain Sensor and an adaptor component that calculates the wiping requests, considering output from `ExteriorLight` and other environmental and vehicle-related information.
`WasherFluidTank` senses the Level of the Washer Fluid container(s), it is instanciated two times to represent the front and rear washer tank.
`EnableDisableWiperWasher` is an adaptation component that manages the enabling of wiper washer functionality according to vehicle specific needs.
`HMI` contains all sensor (and switch) components that detect the drivers demands related to WiperWasher.
`WiperWasherManager` provides the core functionality.
`FrontWasher` is an actuator component. It contains all funtionality for the washing of the windshield.

`FrontWiper` is an actuator component. It provides all the functionality needed for the wiping of the windshield.
`NozzleHeater` is an actuator component. It contains all functionality to heat the headlamp nozzle, including tubes. This component type is instantiated several times to represent the necessary heater for front and rear washer and for the head lamp washer.
`Washer` is an actuator component. It provides all the functionality needed for washing. It is reused to represent the front and rear washers, as well as the headlamp washer.
`Wiper` is an actuator component. It provides all the functionality needed for the wiping of the rear window. It is instanciated to represent the front and rear ashers.

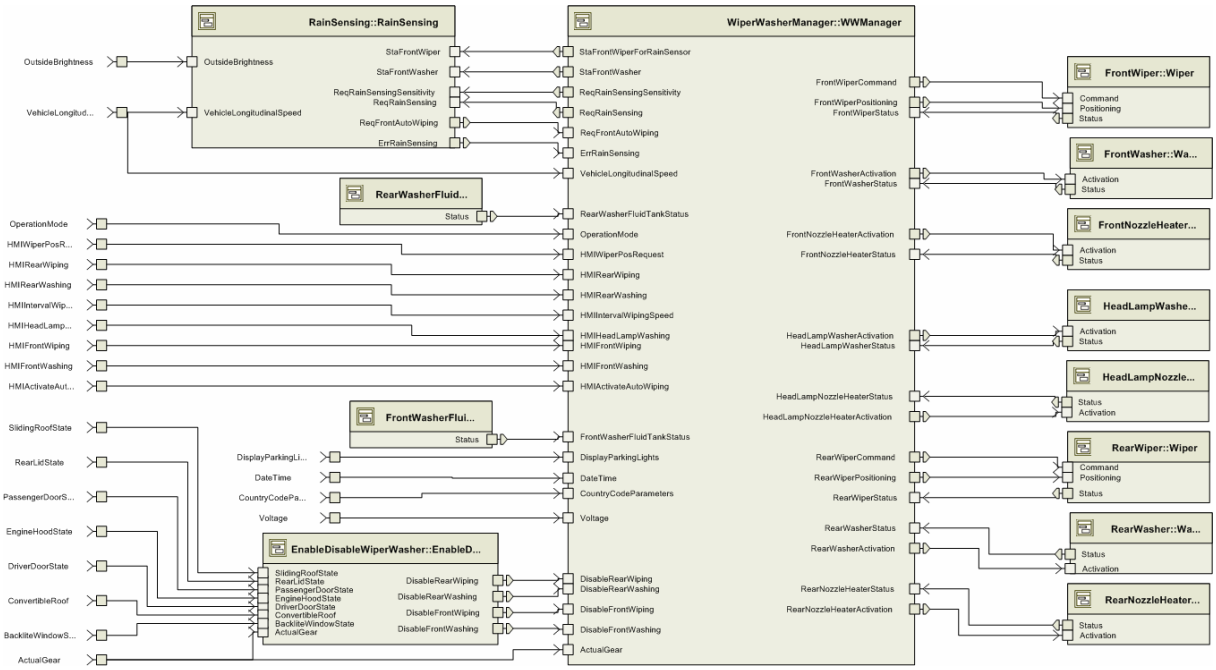


Fig 2.3-1 shows the SW-Cs mentioned above and the software components contained. Data flows are indicated by arrows.

4.1.1 Rain sensing

This section explains the SW-component rain sensing. The following figure gives insight into the decomposition of the SW-composition rain sensing.

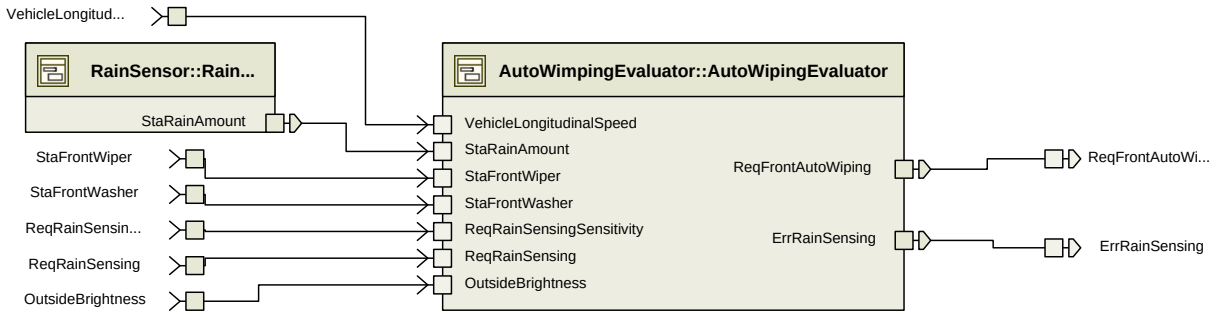


Fig 2.4-1 shows the SW-Cs mentioned above and the software components contained. Data flows are indicated by arrows.

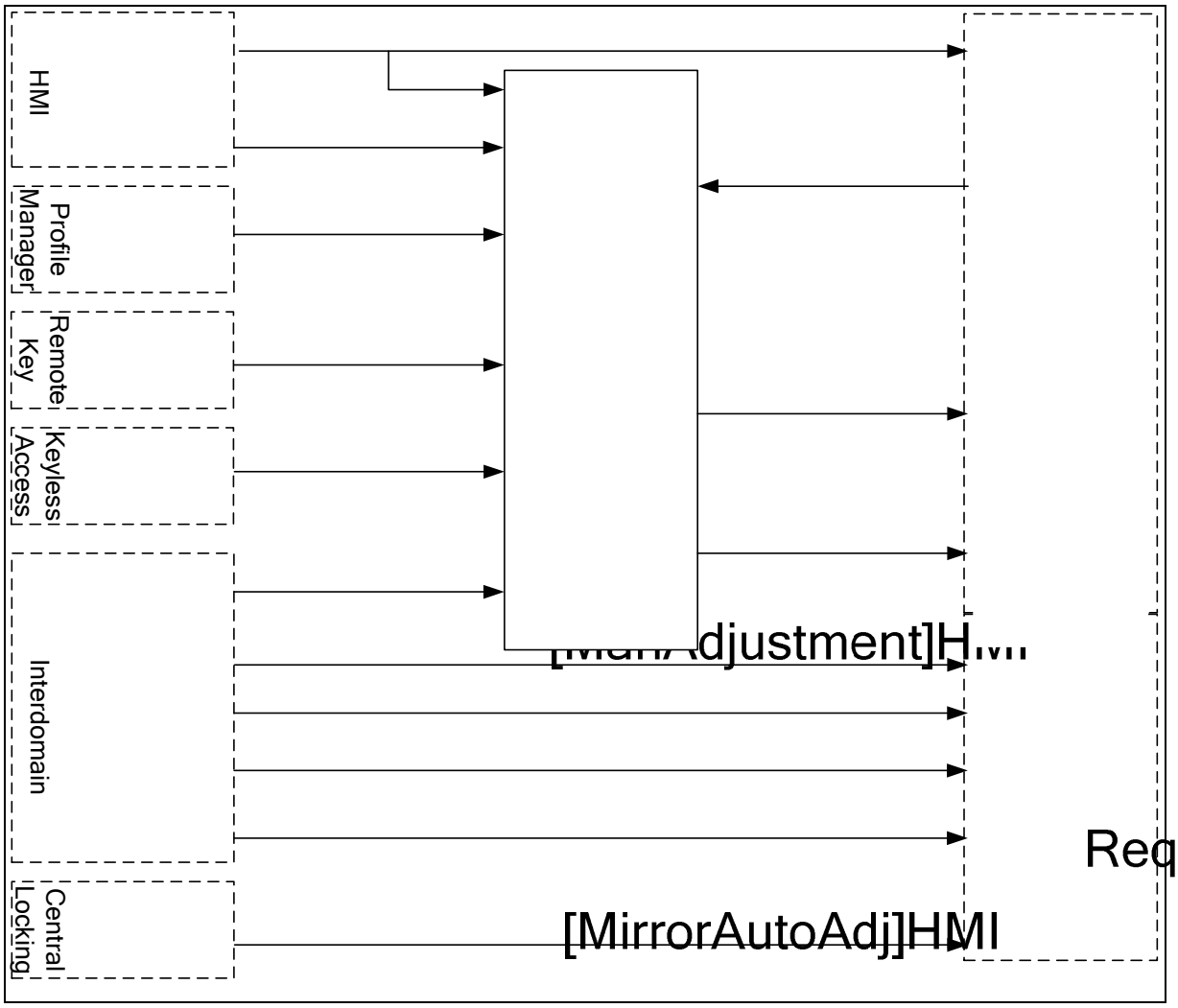
Rain sensing can be split up to the basic `RainSensor`, which delivers some basic information about the detected rain that can be used to calculate / evaluate the need for automatic front wiping.
In the above figure this information is exchanged via the port `StaRainAmount`. As this information is strongly dependent on the used technology, the type of the data-element is currently not binding.
`AutoWipingEvaluator` calculates/ evaluates the request for automatic front wiping. It takes several additional environmental informations into account, including temperature, brightness, speed, wiper/washer information.

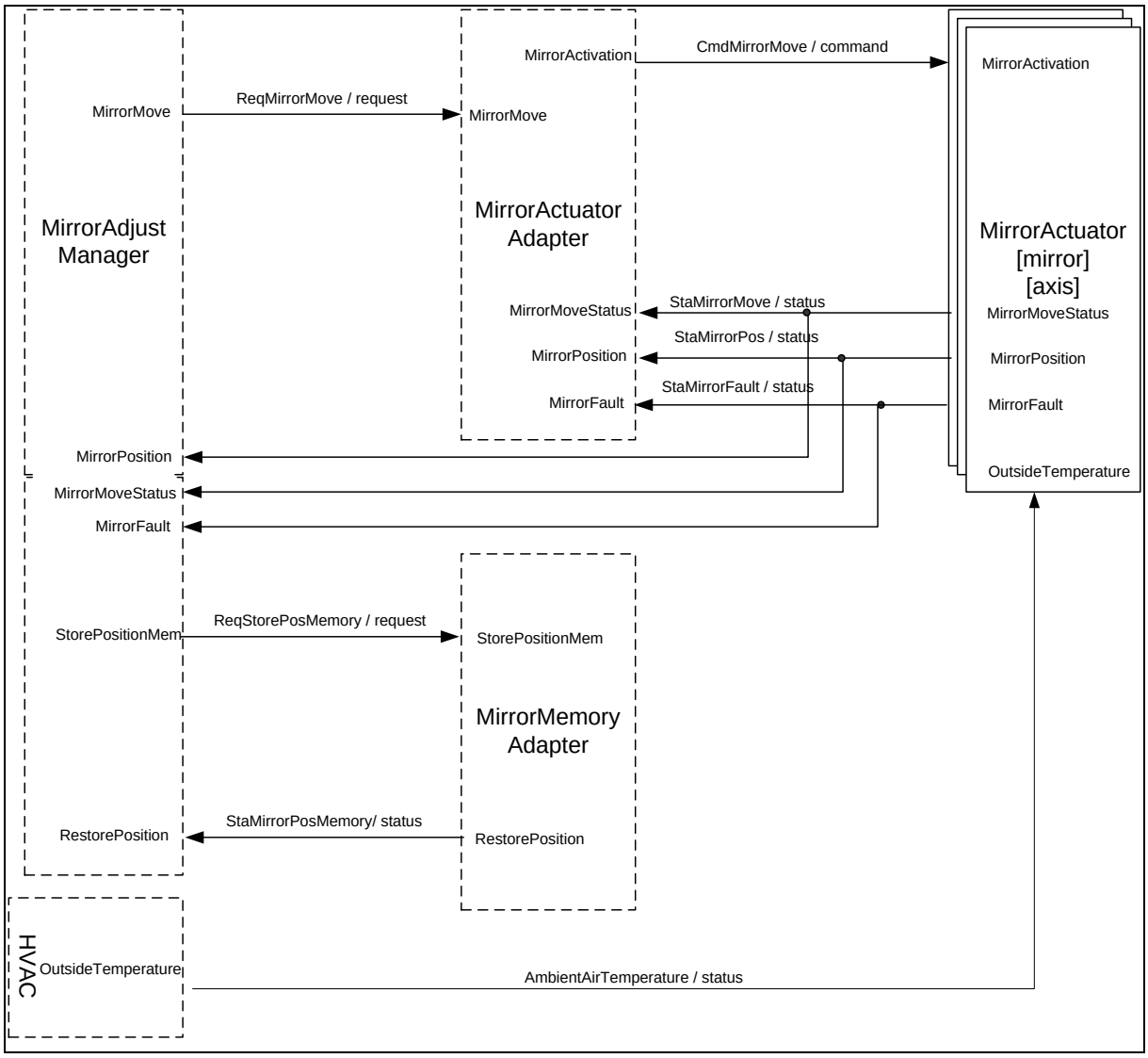
4.2 Mirror Adjustment & Tinting

The mirror adjustment and tinting controls the manual and automatic adjustment of the exterior or interior mirrors of the car, respectively, the tinting level of the mirrors based on the outside brightness intesity.
The functionality has two independent decompositions, one for adjustment and one for tinting. The link between them is the mirror positions and it's movement status.
The functionality needs information from other body domain SW-Cs and also inter-domain information.

4.2.1 Mirror Adjustment

Components related to MirrorAdjustment are bundled in several Blocks.


HMI
Profile Manager
Remote Key
Keyless Access
Interdomain
Central Locking
[MirrorAutoAdj]HMI



Inside MirrorAdjustManager the core functionality is implemented.

HMI contains the Mirror Manual Adjustment, detecting the user's demands for manual adjustment and the Mirror Auto Adjustment, the panel with the memory buttons. It detects the user's demands to store and recall mirror positions.

The Profile manager uses information from the key transponder, keyless access and remote key to generate the current driver profile identification.

The Remote key indicates a user request from the remote key and this may trigger the Auto adjust user request adapter to perform a store or recall.

The Keyless access indicates a user request from keyless access and this may trigger the Auto adjust user request adapter to perform a store or recall.

Inter domain encapsulates SW-Cs that exchange cross-domain information such as selected gear position, operational mode or energy management. Through Inter Domain Interfaces inter-domain status information is exchanged.

Central locking status is received from the Central Locking component.

The Auto adjust user request adapter combines and prioritizes all the information from above and requests mirror movements and storing of memory positions.

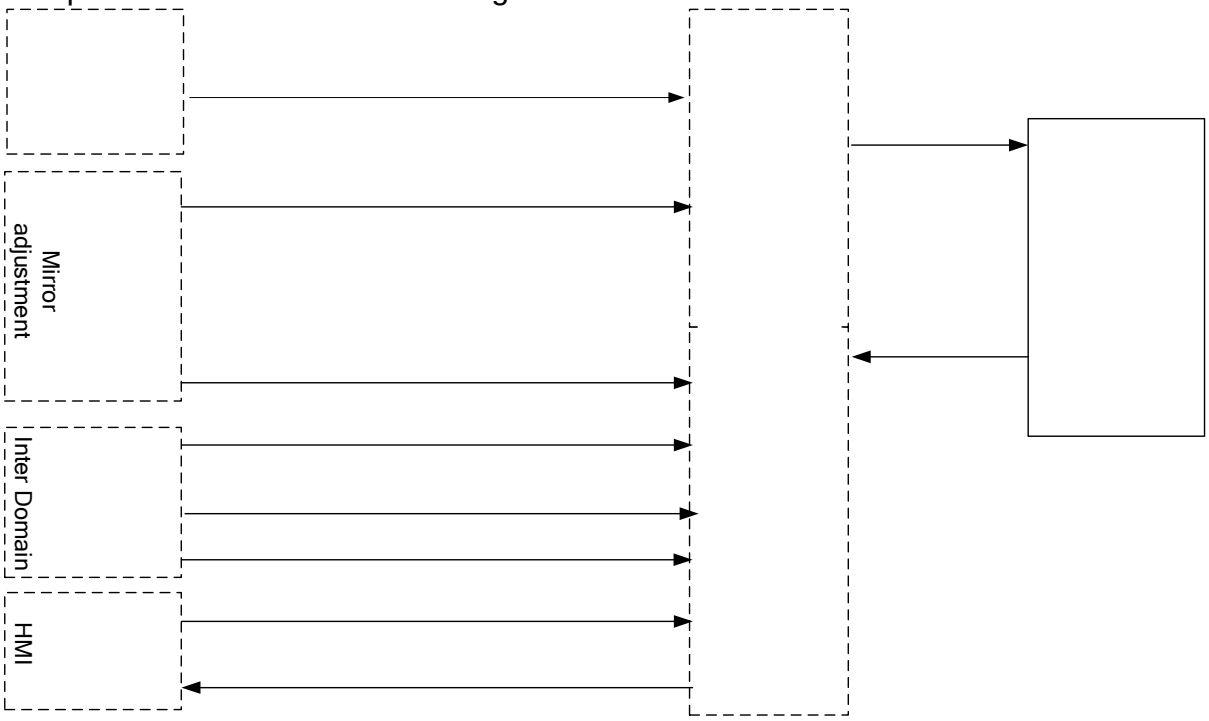
Mirror actuator adapter knows about the electrical and mechanical constraints of each single axes of the mirror. It converts requests from the Mirror adjust manager to commands to each single axes (actuators). It compares the current position of the mirror axis with the desired position.

Mirror memory adapter is responsible for the actual memory position storage and recall. This may be local or global storage depending on the personalization control strategy. External temperature information is used to compensate storage and re-storage of mirror positions.

[mirror][axis]actuator is the software representation of a mechanical actuator. It is able to move one axis of the mirror to a certain direction. It also detects the position of the axis.

4.2.2 Mirror Tinting

Components related to Mirror tinting are bundled in several Blocks.



Inside Tinting Manager the core functionality is implemented.

Based on the information of the Environment Sensing Component, the Tinting Value Manager Component calculates the desired tinting value using the outside brightness information.

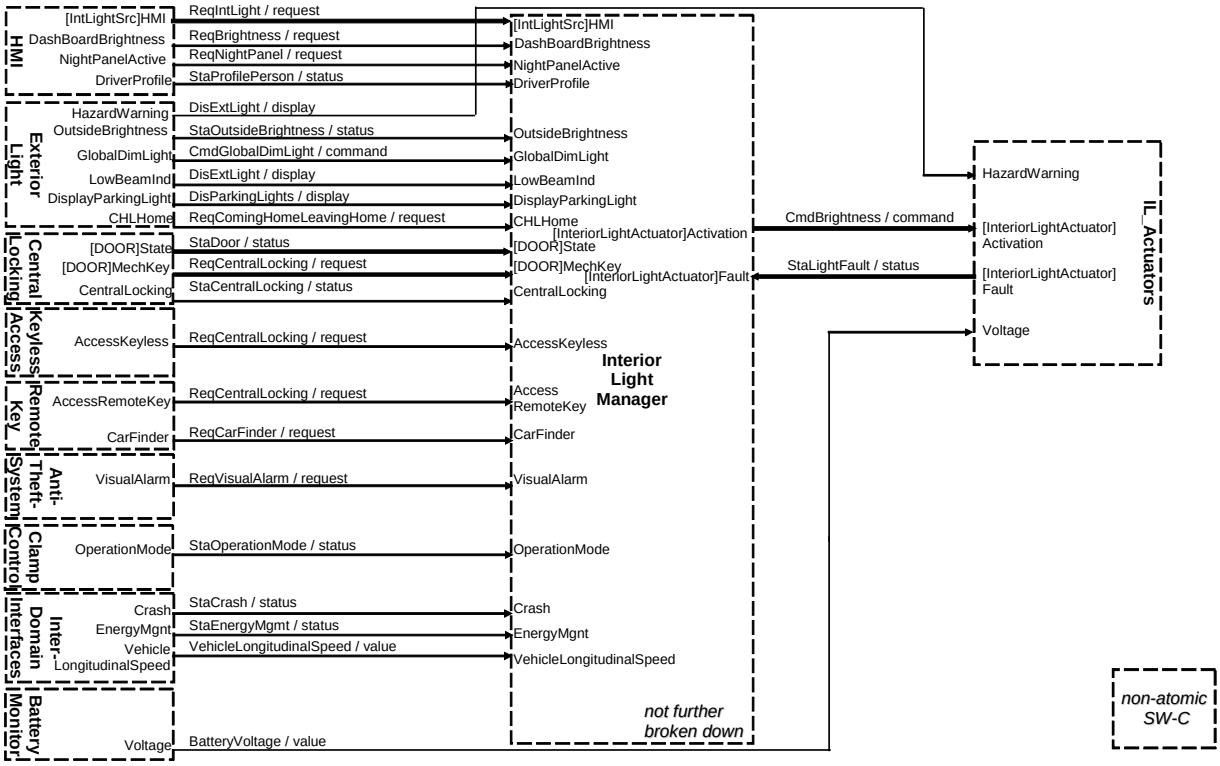
The Inter domain, HIM and Mirror Adjustment Component provide information about the necessity of automatic tinting.

The Mirror Tinting Actuator executes the demands of the Tinting Manager and delivers Status Information back to the Manager.

4.3 Interior Light

The `InteriorLight` software component group controls the interior light functionality of a car. `InteriorLight` receives driver wishes, senses the environment and controls the behavior of the interior light sources of a car.

Components related to `InteriorLight` are bundled in several Blocks.



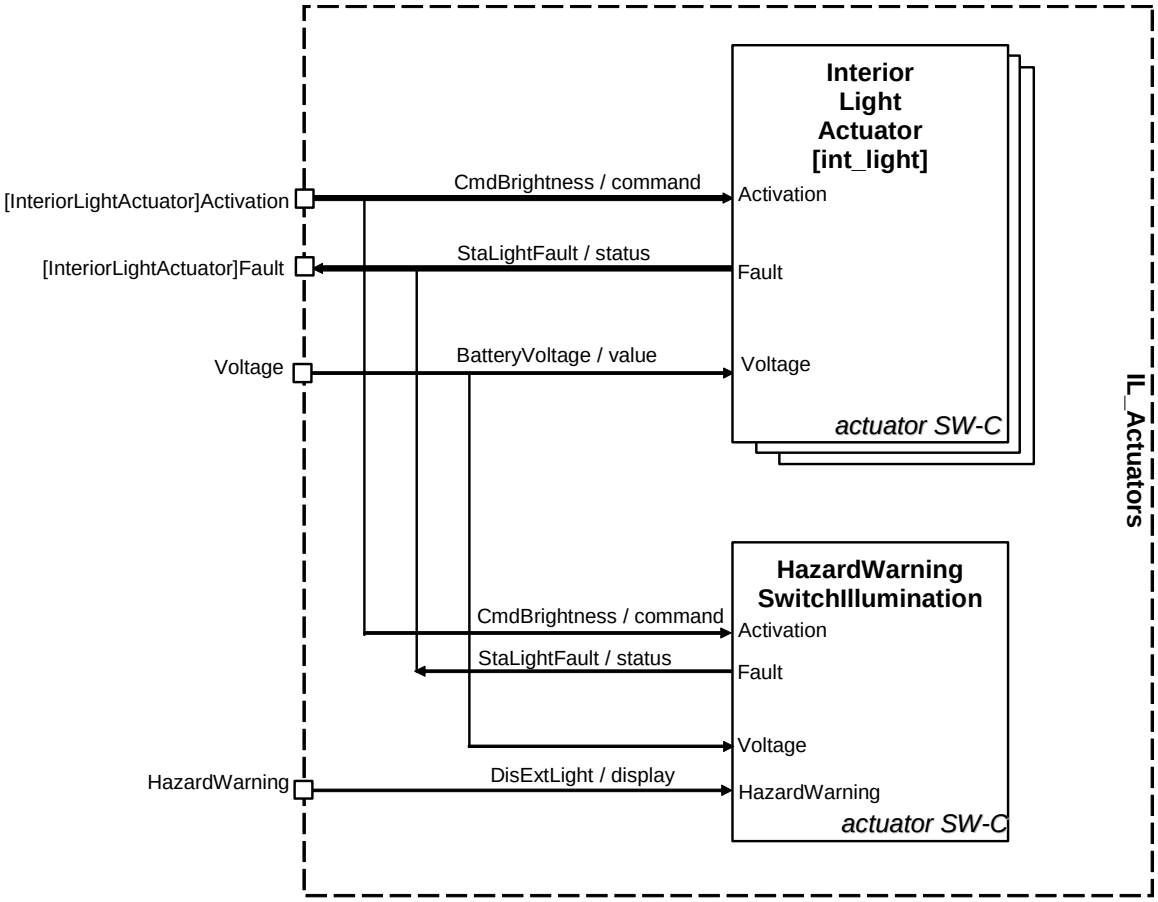


Fig 3.1-1 shows the decomposition of interior light architecture. Arrows indicate data flows.

Remark: This decomposition is used for explanation of standardized interfaces

4.3.1 InteriorLightManager

Inside `InteriorLightManager` the core functionality is implemented (not further broken down).

4.3.2 Interior Light Sensors (HMI)

`HMI` contains all sensor components that detect the drivers demands related to interior light. Please note that “HMI” is used only for non-normative description.

For further decomposition see Fig 3.1-2, whereas usually several SW-C’s InteriorLightSensor[int_light] exist.

The purpose of the Sensor SW-C’s (HMI) is to ensure (as far as needed):
- filtering (debouncing) e.g. calculation of stable output values e.g. by debouncing
- conditioning e.g. calculation of average value
- plausibility check as basis for failure detection (implausible)
- failure detection (but no error-handling/ error-compensation) e.g. valid/not valid

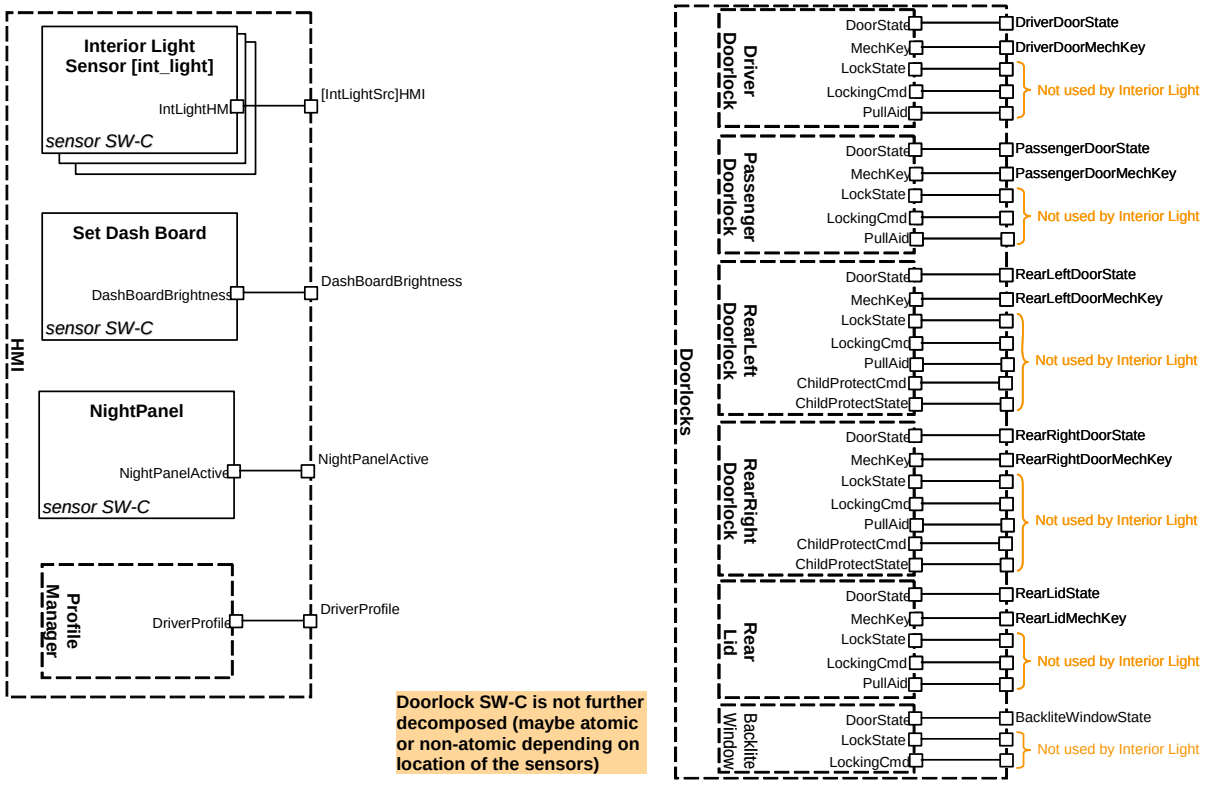


Fig 3.1-2 shows the further Decomposition of HMI & Doorlocks.

4.3.3 CentralLocking (including Doorlocks) / Remote Key / Keyless Access

Out of `Doorlocks` the information is generated whether the corresponding door is open or closed as well as information on mechanical door lock(s). For further details see also separate chapter Central Locking.

Driver requests to lock or unlock the car might also be provided by `Remote Key` or `Keyless Access`. Interior light uses only exterior requests (i.e. issued outside of the car) and not internal requests (such as emergency locking requests).

Note, that interior light may also use the actual door locking status for certain use cases (auto-relocking event starts interior light activity). Reflecting the lock state to the driver is not an intended feature of this standardized interior light.

`Remote Key` also delivers requests for “carfinder” features of the Interior Light.

4.3.4 Exterior Light

Other blocks like `ExteriorLight` provide some information as well as Environment Sensing (e.g. Light Sensor) which provides information about the outside brightness.

For further details on the above mentioned functionalities see also separate chapter Exterior Light.

4.3.5 Anti-Theft-System

One interfaces is provided also by this body and comfort functionalitiy. For further details on Anti-Theft-System see according chapter (to be included in later releases).

4.3.6 ProfileManager

It is supposed only a single source coordinates profile Ids from certain personalization sources (remote key, keyless etc.). The `ProfileManager` is NOT in scope of this document.

4.3.7 Clamp Control

The status of ignition key is provided by `Clamp Control`.

4.3.8 Inter-Domain Interfaces

Through `Inter-Domain-Interfaces` inter-domain status information is exchanged.

4.3.9 Battery Monitor

`Battery Monitor` provides information about Battery Voltage.

4.3.10 Actuators

`InteriorLightActuator[int_light]` (including `HazardWarningSwitchIllumination`): All interior light actuators are described by the `LightSource[int_light]`components where [int_light] is a placeholder for individual light sources (e.g. CabinLightFnCe as Cabinlight “Front-Middle”) or a group of light sources (e.g. reading_lights). See next chapter for a list of possible effective names.

It might be decomposed into a light actuator type independent and a light actuator type dependent component, where light actuator types include conventional bulbs, LEDs or other decompositions, which allow the switching of lights on a more “generic” level.

The actuator also covers low level functionalities (e.g. measures to prevent flickering due to voltage variations) as well as over-/under voltage compensation. Also hardware dependencies are considered (e.g. is PWM possible for corresponding actuator). Status feedback is generated (only operational/defective; no detailed diagnosis). Actuator SW-Cs do not consider system restrictions (e.g. restrictions due to mechanical reasons regarding activation of several actuators). This should be task of a separate Actuator-Adapter SW-C.

Restrictions to Interior Light Actuator: Actuator for IntLightGroup0..15 is allowed only if all light sources within the group are
1) not part of any other light group
2) not directly used by another light actuator
Same applies for groups like CabinLights, TrunkCompartmentLights, FootwellLights etc.

4.3.11 List of interior light sources – groups of light sources

The list below defines individual light sources (like CabinLightFnRi) as well as functional groups (e.g. CabinLights) as well as virtual groups (see IntLightGroup0..15):

CabinLights
- CabinLightFnRi (front_right)
- CabinLightFnLe (front_left)
- CabinLightFnCe (front_center e.g. in the middle)
- CabinLightMidRi (middle_right)
- CabinLightMidLe (middle_left)
- CabinLightMidCe (middle_center)
- CabinLightReRi (rear_right)
- CabinLightReLe (rear_left)
- CabinLightReCe (rear_center)

TrunkCompartmentLights
- TrunkCompartmentLightCe (light_middle)
- TrunkCompartmentLightLoRi (lower_right)
- TrunkCompartmentLightLoLe (lower_left)
- TrunkCompartmentLightUpRi (upper_right)
- TrunkCompartmentLightUpLe (upper_left)

FootwellLights
- FootwellLightFnLe (front_left)
- FootwellLightFnCe (front_middle)
- FootwellLightFnRi (front_right)
- FootwellLightMidLe (middle_left)
- FootwellLightMidCe (middle_middle)
- FootwellLightMidRi (middle_right)
- FootwellLightReLe (rear_left)
- FootwellLightReCe (rear_middle)
- FootwellLightReRi (rear_right)

AshtrayLights
- AshtrayLightFnLe (front_left)

AshtrayLightFnCe (front_middle)
AshtrayLightFnRi (front_right)
AshtrayLightMidLe (middle_left)
AshtrayLightMidCe (middle_middle)
AshtrayLightMidRi (middle_right)
AshtrayLightReLe (rear_left)
AshtrayLightReCe (rear_middle)
AshtrayLightReRi (rear_right)

ReadingLights
ReadingLightFnLe (front_left)
ReadingLightFnCe (front_middle)
ReadingLightFnRi (front_right)
ReadingLightReLe (rear_left)
ReadingLightReCe (rear_middle)
ReadingLightReRi (rear_right)
ReadingLight3rdRowLe (middle_left)
ReadingLight3rdRow Ce (middle_middle)
ReadingLight3rdRow Ri (middle_right)

CurbLights
CurbLightFnLe (front_left)
CurbLightFnRi (front_right)
CurbLightMidLe (middle_left)
CurbLightMidRi (middle_right)
CurbLightReLe (rear_left)
CurbLightReRi (rear_right)

VanityLights
VanityLightFnLe (front_left)
VanityLightFnRi (front_right)
VanityLightMidLe (middle_left)
VanityLightMidRi (middle_right)
VanityLightReLe (rear_left)
VanityLightReRi (rear_right)

CigaretteLighterIlluminationLights
CigaretteLighterIlluminationLightFnLe (front_left)
CigaretteLighterIlluminationLightFnCe (front_middle)
CigaretteLighterIlluminationLightFnRi (front_right)
CigaretteLighterIlluminationLightMidLe (middle_left)
CigaretteLighterIlluminationLightMidCe (middle_middle)
CigaretteLighterIlluminationLightMidRi (middle_right)
CigaretteLighterIlluminationLightReLe (rear_left)
CigaretteLighterIlluminationLightReCe (rear_middle)
CigaretteLighterIlluminationLightReRi (rear_right)
CigaretteLighterIlluminationLightTrLe (trunk_left)
CigaretteLighterIlluminationLightTrRi (trunk_right)

DoorSillIlluminationLights
DoorSillIlluminationLightFnLe (front_left)
DoorSillIlluminationLightFnRi (front_right)
DoorSillIlluminationLightMidLe (middle_left)
DoorSillIlluminationLightMidRi (middle_right)

DoorSillIlluminationLightReLe (rear_left)
DoorSillIlluminationLightReRi (rear_right)

DoorOpenerIlluminationLights
DoorOpenerIlluminationLightFnLe (front_left)
DoorOpenerIlluminationLightFnRi (front_right)
DoorOpenerIlluminationLightMidLe (middle_left)
DoorOpenerIlluminationLightMidRi (middle_right)
DoorOpenerIlluminationLightReLe (rear_left)
DoorOpenerIlluminationLightReRi (rear_right)

SafetyBeltLockIlluminationLights
SafetyBeltLockIlluminationLightFnLe (front_left)
SafetyBeltLockIlluminationLightFnCe (front_middle)
SafetyBeltLockIlluminationLightFnRi (front_right)
SafetyBeltLockIlluminationLightMidLe (middle_left)
SafetyBeltLockIlluminationLightMidCe (middle_middle)
SafetyBeltLockIlluminationLightMidRi (middle_right)
SafetyBeltLockIlluminationLightReLe (rear_left)
SafetyBeltLockIlluminationLightReCe (rear_middle)
SafetyBeltLockIlluminationLightReRi (rear_right)

GloveCompartmentLight
EngineCompartmentLight
IgnitionSwitchIllumination
HazardWarningSwitchIllumination
ExteriorLightSwitchIllumination

These groups of interior light interfaces are defined (virtual interior light interfaces):
IntLightGroup0,
IntLightGroup1,
…
IntLightGroup15,

Each group 0..15 can consist out of any number of above mentioned interior light sources. This is configurable per implementation.

Examples for above mentioned groups are:
SwitchIlluminationGroupAccessory
SwitchIlluminationGroupIgnitionOn
SwitchIlluminationGroupExteriorLightOn
DoorAjarFrontLeftGroup
DoorAjarFrontRightGroup
DoorAjarRearLeftGroup
DoorAjarRearRightGroup

As a further example for the application of an “IntLightGroup” there is the „Battery-Saver Feature“ for deactivation of inadvertent loads of interior light (example glove box: lamp(s) might be hardwired activated by a switch. If left open unintentionally a separate actuator could be provided for deactivation of all these hardwired lamps after a certain timeout after ignition off).

The groups of light sources mentioned above (e.g. IntLightGroup0..15) may be configured by means of “Configuration-Bitfields” of type t_LightGroupConfiguration whereas each position corresponds to a certain interior light source (see typedef-documentation). This means that this information might be stored in NVRAM and accessible through calibration and/or diagnostic tools. Also possible is providing this information as static input to Interior Light Manager.

4.3.12 Known Deficiencies

Limitations of the MasterTable:
Multiple instances of compositions cannot be handled properly by MasterTable. Therefore only two possible actuator SWC’s (CabinLightFrontMiddle & HazardWarningSwitchIllumination) are listed in the Master Table. As a consequence generated XML coded has most probable to be reworked manually.

Responsibility for illumination of switches:
Item not yet finally decided - current feedback from WPII-10.5: It’s not a problem, when WPII-10.1 defines HMI related components inside own workgroup, when the HMI is closely related to WPII-10.1 domain and have no interaction with other domains. Illumination seems to be such a condidate.

Diagnostics:
Diagnostics Ports are not part of this explanatory description. Nevertheless, diagnostic interfaces/ports are defined (see Master Table). Please note that all ports referencing the interface *DiagnosticMonitor* are client ports.

Not in scope of this document: Diagnostic events of other SW-C’s than Actuator SW-C. This implies that the former ports referencing the interfaces *DiagSensor*, *DiagBrightness* as well as all client ports indicating sensor faults by means of interace *DiagnosticMonitor* are to be defined within HMI.

Further ports to AUTOSAR-Services:
For ports to ECU State-Management as well as COM-Services see according AUTOSAR documentation.

Regarding Cal-Ports there are only implementation specific parameters for Interior Light Manager. Please note that “pure ROM parameters” (like PWM frequency) are not listed here.

For each Interior Light Actuator certain parameters are accessible via “cal-ports” (further details are supposed to be specified by RTE-Team).
Examples are:
- value for “full brightness”: light actuator is set to e.g. 90% of full brightness due to possibility to compensate low voltage.
- ramp characteristics.

HINT - Relationship between Diagnostics and internal state of SW-Cs: The interaction of SW-Cs with the Diagnostic Communication Manager (DCM) takes place exclusively through the corresponding AUTOSAR DCM Service. AUTOSAR Services

are accessible, like any other communication, through interfaces and ports and thus Communication Modes apply. Through the application of Communication Modes to determined RTEEvents (like receiving a data element or an invocation to a function of a client-server interface), it is possible to avoid determined activities under special circumstances. As an example, given a diagnostic service provided by a SW-C and given the SW-C shall not allow any diagnostic call under a special state, we only need to define a ModeDisablingDependency for the corresponding RTEEvent (in this case OperationInvokedEvent) and the undesired state. Is there a call being made for this function, when the SW-C is in the undesired state, the RTEEvent (OperationInvokedEvent) will be discarded and thus the function of the SW-C won't be called. This means that the state of the SW-Cs can be considered by the diagnostics, behavior is up to the application developer. For more information, please check the RTE specification and the AUTOSAR Services Document (DCM Service) of System Team.

4.4 Seat Adjustment

The `SeatAdjustment` software component group controls the seat adjustment functionality of a car. `SeatAdjustment` receives driver wishes, senses the environment and controls the behavior of the `SeatAdjustment` actuators in a car.

4.4.1 Seat Configurations

The following table lists the allowed seat names that are defined for use in AUTOSAR systems. The table also shows the name text that should be used for constructing port and interface references in place of the place holder “[seat]”.

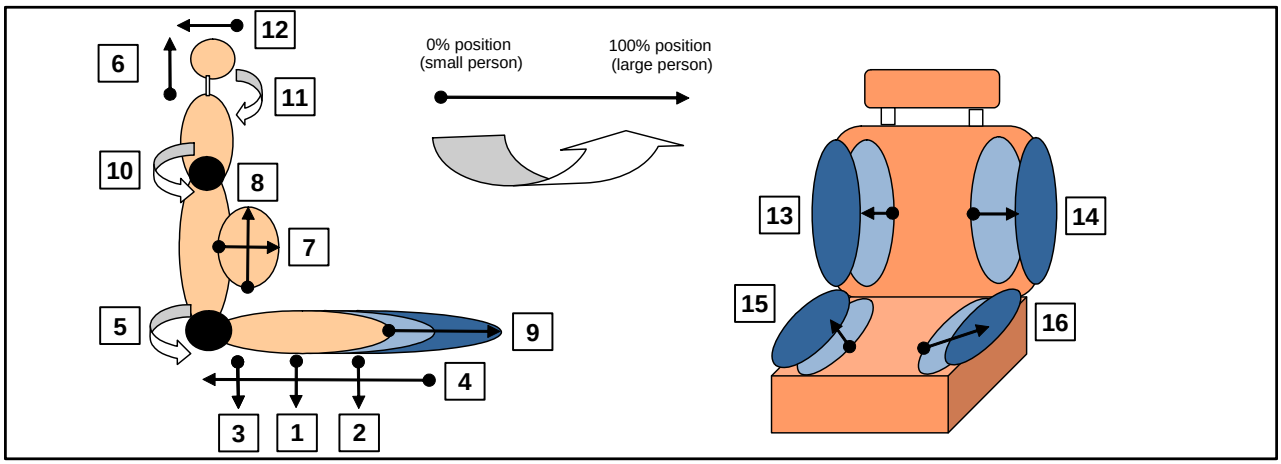
Seat Name	***Reference Text***
Driver Seat	Driver
Front Passenger Seat	PassFr
Front Central Seat	CentreFr
Left Middle Seat	LeftMid
Centre Middle Seat	CentreMid
Right Middle Seat	RightMid
Left Rear Seat	LeftRr
Centre Rear Seat	CentreRr
Right Rear Seat	RightRr

The following table lists the allowed axis names that are defined for use in AUTOSAR systems. The table also shows the name text that should be used for constructing port and interface references in place of the place holder “[axis]”.

Diagram ID	***Axis Name***	***Reference Text***
1	Seat Height	Height
2	Seat Front Height	FrHeight
3	Seat Rear Height	RrHeight
4	Seat Slide	Slide
5	Seat Back Inclination	Incline
6	Headrest Height	HdHeight
7	Lumbar Extend	LumExtend

8	Lumbar Height	LumHeight
9	Seat Cushion Extend	CushExtend
10	Seat Back Top Inclination	TopIncline
11	Headrest Fold	HdFold
12	Headrest Tilt	HdTilt
13	Seat Back Right Bolster	BkRBolster
14	Seat Back Left Bolster	BkLBolster
15	Seat Cushion Right Bolster	CushRBolster
16	Seat Cushion Left Bolster	CushLBolster

The axes are shown on the diagram below to show the sense of the position data.



4.4.2 Seat Adjustment Decomposition

The Seat Adjustment sub-system is shown in Figure 3.1-1. Software Components related to `SeatAdjustment` are described in the following text.

4.4.2.1 HMI

The `HMI` component provides the interface from user operations.

`ReqMassageAdjust` enables the user to choose one or more massage programs from a massage control panel (e.g. a switchpack or touch screen menu) and inform the `[seat]SeatAdjustManager`. There may be specific “massage actuators” in addition to the defined seat axes. These actuators are not named here.

`StaMassage` provides status of the active massage functions back to the HMI to inform the user of the current mode of operation.

`ReqManAdjust` enables the user to activate one or more seat actuator using a manual movement control panel (e.g. a switchpack or touch screen menu) and inform the `[seat]SeatAdjustManager`. This information additionally goes to the `[seat]AutoAdjustUserRequestManager` so that manual operations can be used to over-ride memory adjustment operations.

`ReqAutoAdjust` indicates the user's demands to store and recall seat positions.

`ReqRearEntry` detects the users demand for access to the rear seats. This may result in a front seat moving forward. This feature may also be known as auto glide. Similarly, easy front entry requests are possible; see `ReqFrontEntry`.

4.4.2.2 Doors, KeylessAccess and RemoteKey

The `Central Locking` component provides information about operation of vehicle accesses (doors and locks) that might indicate a need to prepare seats for occupation. The `ProfileManager` uses information from the key transponder, keyless access and remote key to generate the current driver profile identification `StaProfilePerson`.

If the `trigger_source` inside of the interface `StaCentralLocking` indicates a user request from the remote key or keyless entry, this may trigger the `[seat]AutoAdjustUserRequestManager` to perform a store or recall.

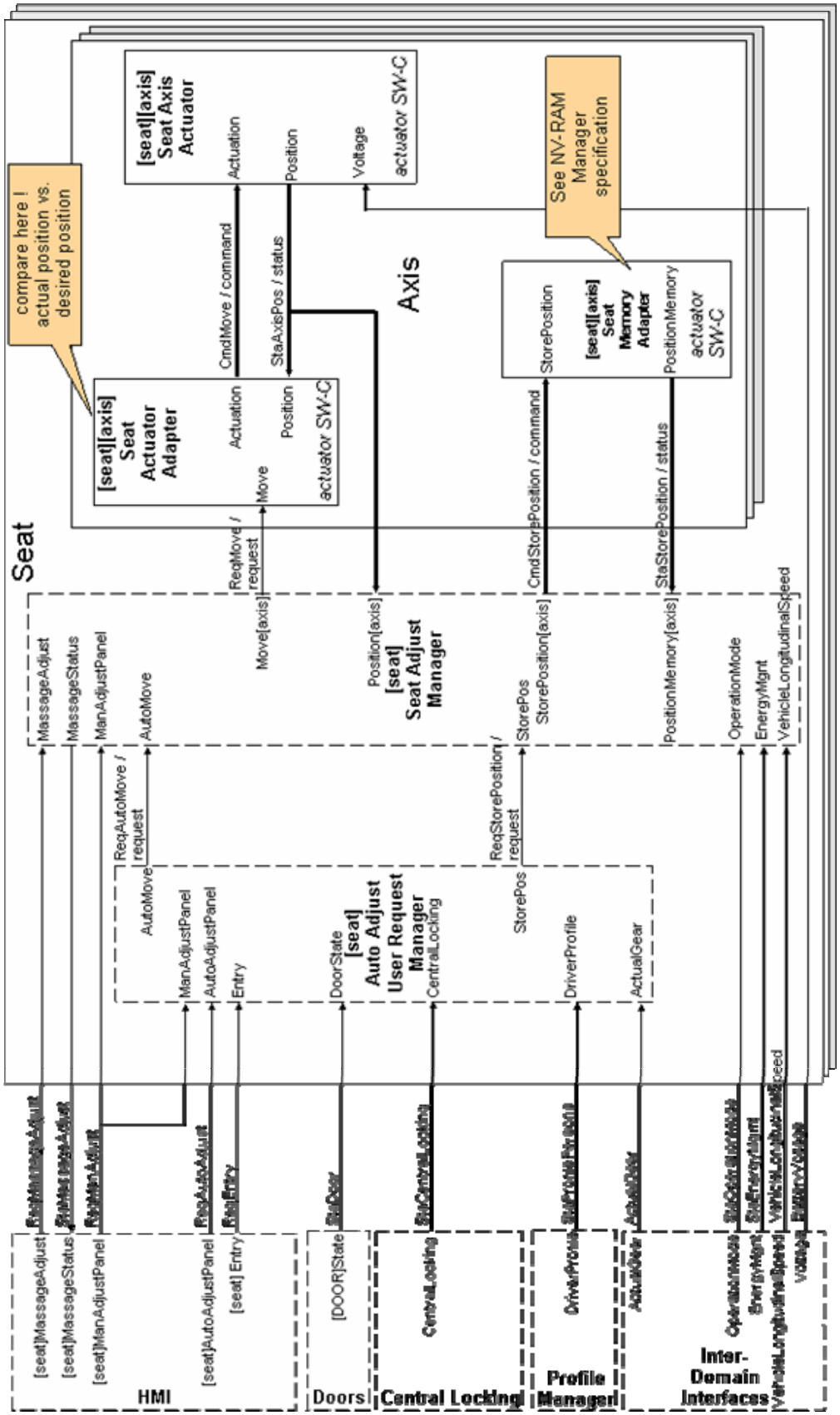


Fig 4.5-1 shows the SW-Cs mentioned above and the software components contained. Arrows indicate data flows.

The CentralLocking indicates a user request from central locking system and this may trigger the [seat]AutoAdjustUserRequestManager to perform a store or recall.

The StaDoor indicates whether the door associated with the seat has been opened or closed and this can be used to trigger seat movements or to prevent movement.

4.4.2.3 Inter-DomainInterfaces

Inter-DomainInterfaces encapsulates interfaces to SW-Cs that exchange cross-domain information such as vehicle speed, vehicle longitudinal and lateral accelerations, operational modes or energy management status. Inter-domain status information is exchanged through Inter-Domain-Interfaces.

4.4.2.4 AutoAdjustUserRequestManager

There is a [seat]AutoAdjustUserRequestManager associated with each seat that requires seat adjustment. The [seat]AutoAdjustUserRequestManager combines and prioritizes all the information from above and requests seat movements and storing of memory positions.

4.4.2.5 ActuatorAdapter

There is a [seat][axis]ActuatorAdapter associated with each axis in every seat that requires seat adjustment control. [seat][axis]ActuatorAdapter knows about the electrical and mechanical constraints of a single axis of the seat. It converts requests from the [seat]SeatAdjustManager into commands to a single axis (actuator). It compares the current position of the seat axis with the desired position.

4.4.2.6 SeatAxisActuator

There is a [seat][axis]SeatAxisActuator associated with each axis in every seat that requires seat adjustment control. [seat][axis]SeatAxisActuator is the software representation of a mechanical actuator. It is able to move one axis of the seat in a certain direction. It also detects the position and speed of the axis.

4.4.2.7 SeatMemoryAdapter

There is a [seat][axis]SeatMemoryAdapter associated with each axis in every seat that requires seat adjustment control.
[seat][axis]SeatMemoryAdapter is responsible for the actual memory position storage and recall. This may be local or global storage depending on the personalization control strategy.

4.4.2.8 SeatAdjustManager

Finally, the `[seat]SeatAdjustManager` is the core of the seat adjustment implementation. There is a `[seat]SeatAdjustManager` associated with each seat that requires seat adjustment. This component coordinates all the various requests and triggers into a specific control strategy for a seat. It also ensures that seat adjustment is carried out safely and in accordance with feature functional priorities and behaviors.

4.4.3 Personalization Issues

It is expected that seat adjustment can be personalized.

Known use cases for personalization of the seat adjustment functionality are, for example:

- Store the preferred seat position of the driver;
- Enable/disable certain seat adjustment functionality (e.g. easy entry).

4.4.4 Design Rationale

This design standard is the AUTOSAR-architecture for seat adjustment. It provides a decomposition into SW-Cs and a list of standardized interfaces related to seat adjustment.

The decomposition is limited in granularity to sensor components, adapter components, the core functionality and actuator components. It is not intended to focus only on event driven implementations and the standardization should be also open for “cyclic implementations” sending out information periodically (therefore some interfaces contain an “idle” value by intent).

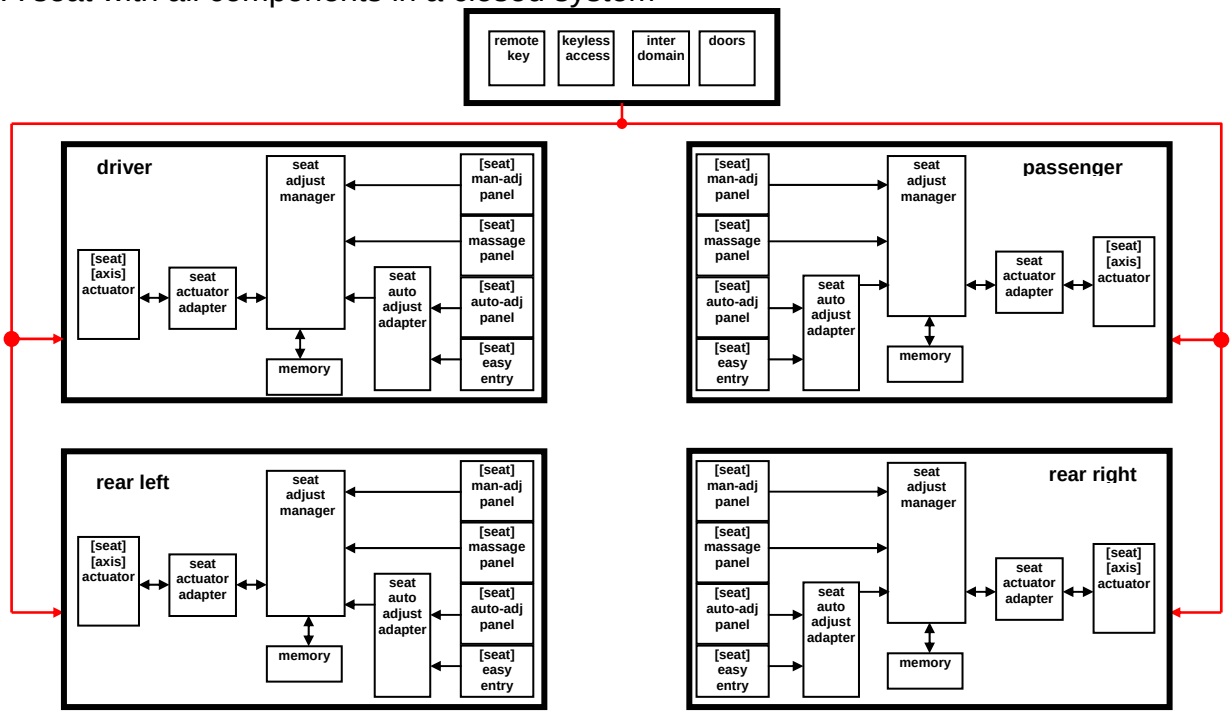
Additionally, all ports are described together with AUTOSAR agreed data qualities. Invalidation (where needed) is defined as in-band invalidation. Init values are specified where appropriate e.g. “off”, “idle”, “undefined”, “unknown” … Where standardization is in-appropriate, recommendations may be provided.

The decomposition in this state is good enough to gather a common understanding of interfaces for standardization. A finer granularity would not provide greater ease of standardization, as noted during exterior light standardization.

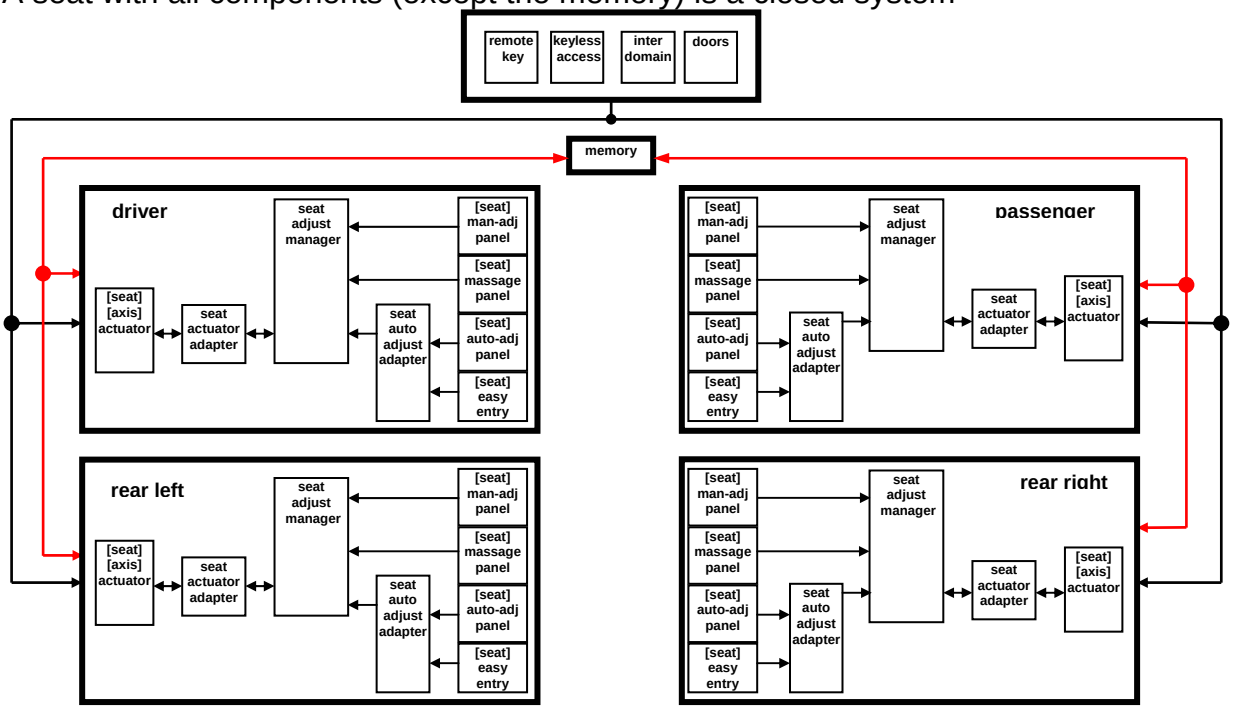
By intent the decomposition does not get to atomic SW-Cs but to “purchasable” SW-Cs (although sensor/actuator SW-Cs are realized as AUTOSAR compositions for formal reasons). These SW-Cs will be obtained as a unit so that all the internal (and hence not AUTOSAR standardized) interfaces are controlled by a single vendor, even if there are SW-Cs within the bought unit that reside on different ECUs. All interfaces between SW-Cs from different vendors should be standardized.

4.4.5 Architectures

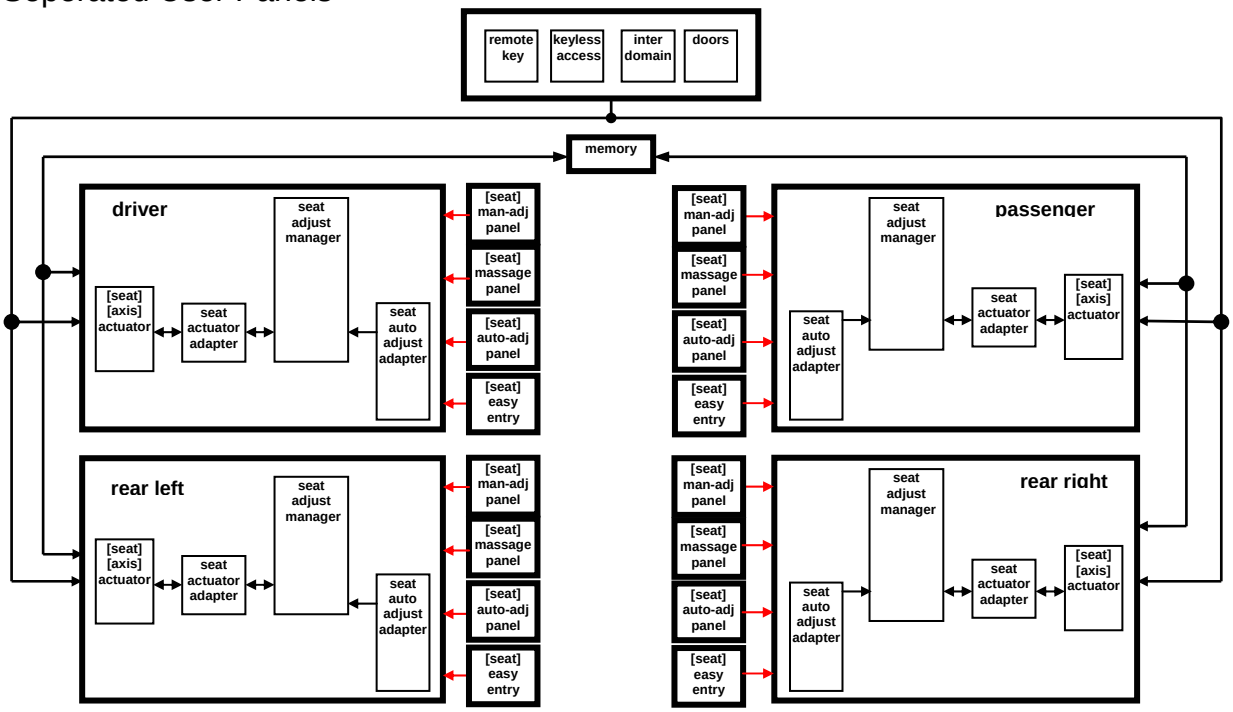
A seat with all components in a closed system



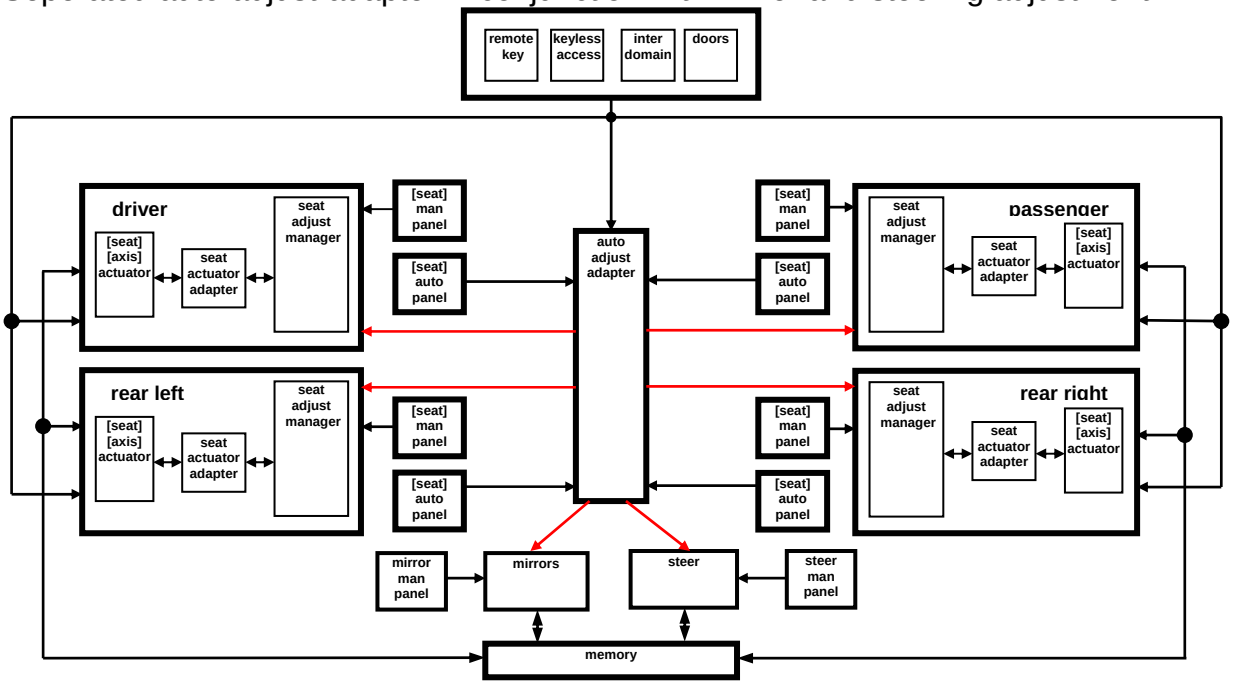
A seat with all components (except the memory) is a closed system



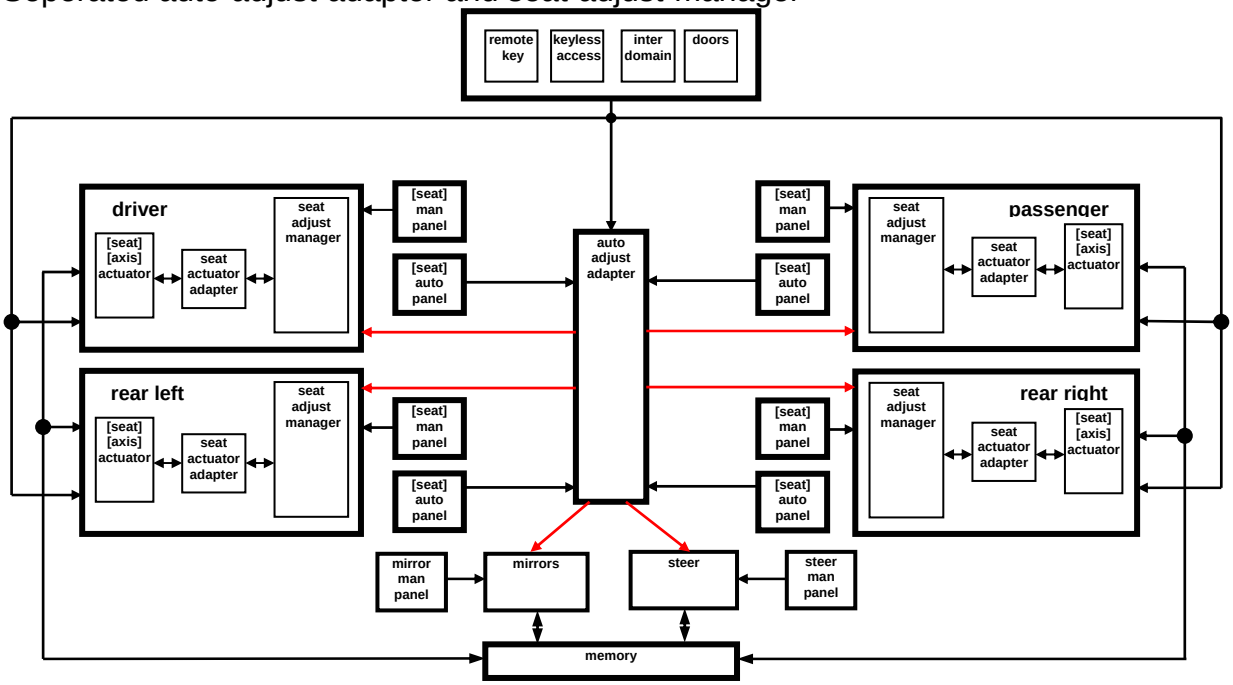
Seperated User Panels



Seperated auto-adjust-adapter in conjunction with mirror and steering adjustment



Seperated auto-adjust-adapter and seat-adjust-manager

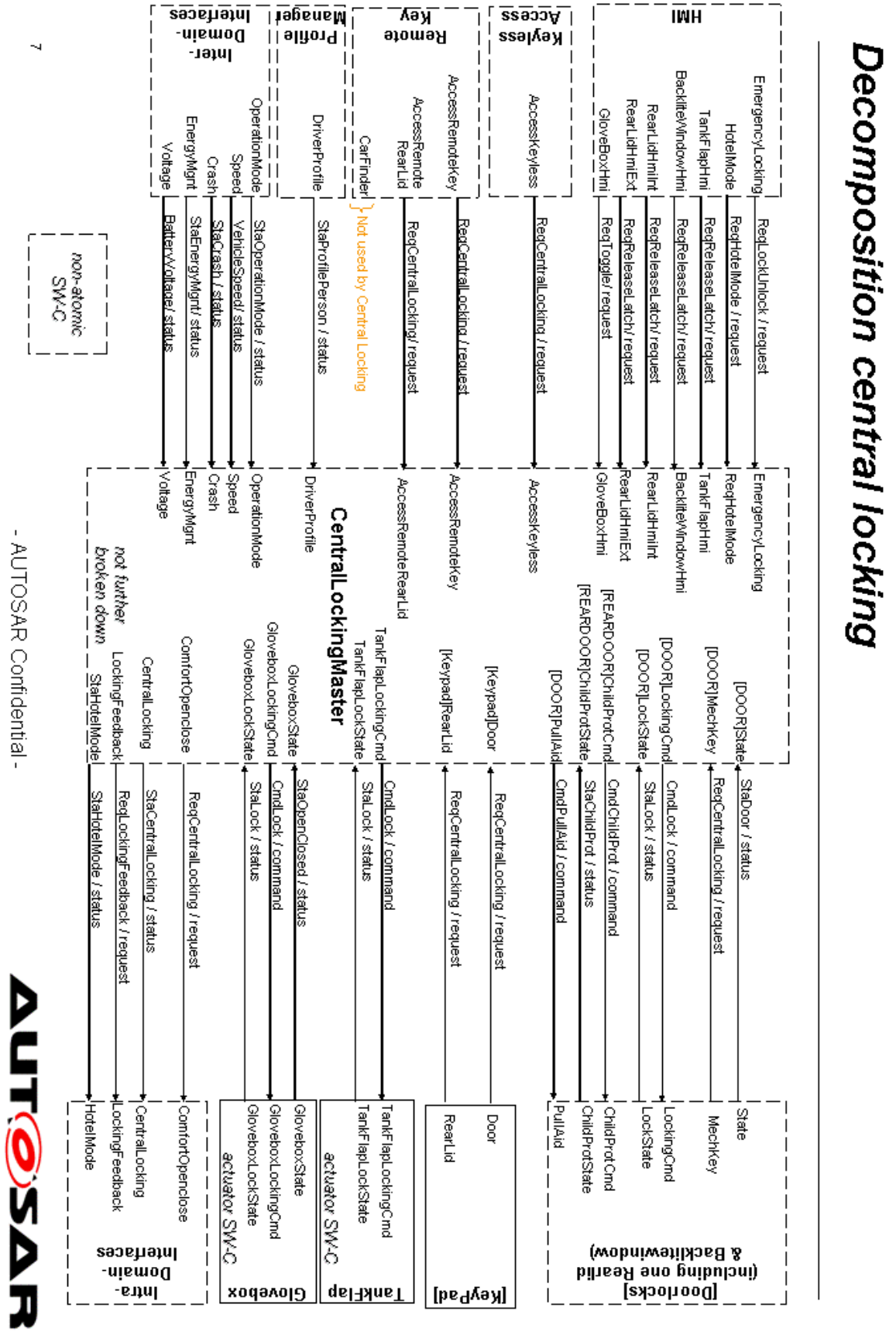


4.5 Central Locking

The `CentralLocking` software component group controls the central locking functionality of a car. `CentralLocking` receives driver wishes, senses the environment and controls the behavior of the central locking sources of a car.

4.5.1 Central Locking Decomposition

Components related to `CentralLocking` are bundled in several blocks:



Pease note that diagnostic interfaces are not part of the graphics.

4.5.2 Explanation of SWCs

CentralLockingMaster
Inside CentralLockingMaster the core functionality is implemented (not further broken down).

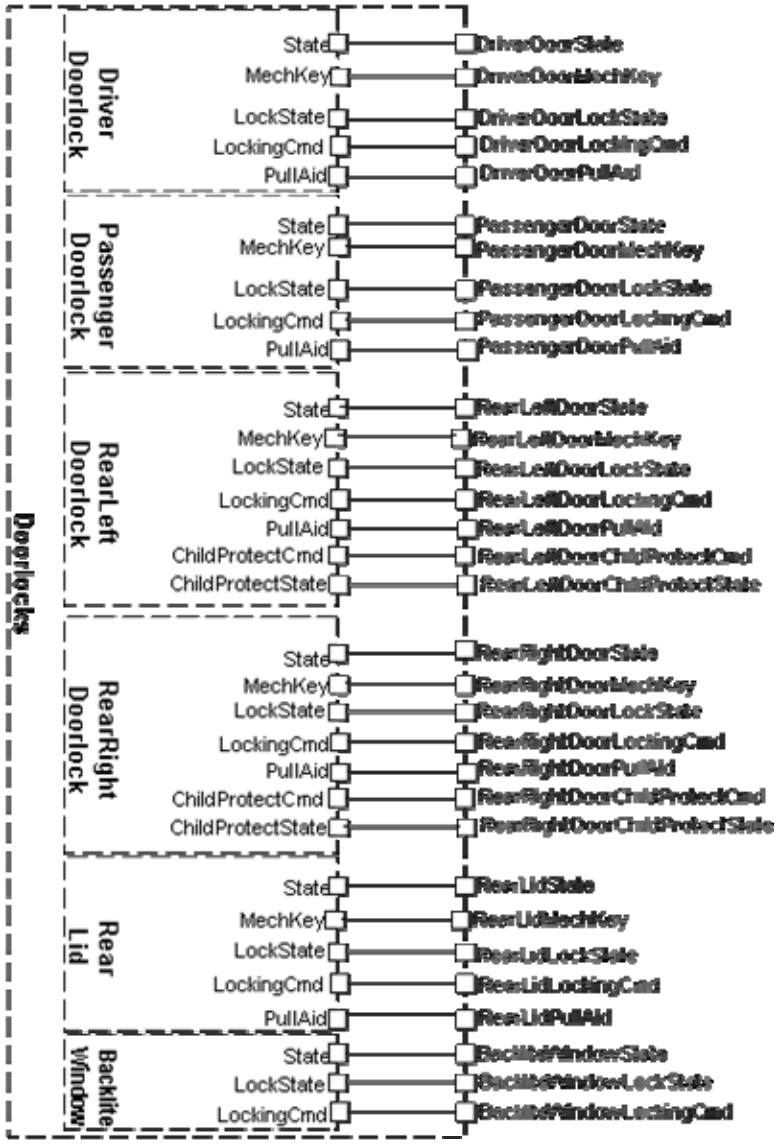
HMI
HMI contains all other sensor components that detect the drivers demands related to central locking. Please note that “HMI” is used only as non-normative description.

RemoteKey / Keyless Access
Driver requests to access the car might be provided by RemoteKey or Keyless Access System (including profile information for personalization).

Doorlocks
Out of the doorlocks the information of the doorlock status is generated as well as information on mechanical keys.

Lock-commands as well as commands for pull aids and setting the child-protection-mechanism are also sent to the doors. Same applies for RearLid and BackliteWindow. See also “mechanical decomposition door” and “mechanical events vs. standardized interfaces” (see below).

Doorlock SW-C is not further decomposed (maybe atomic or non-atomic depending on location of the sensors)

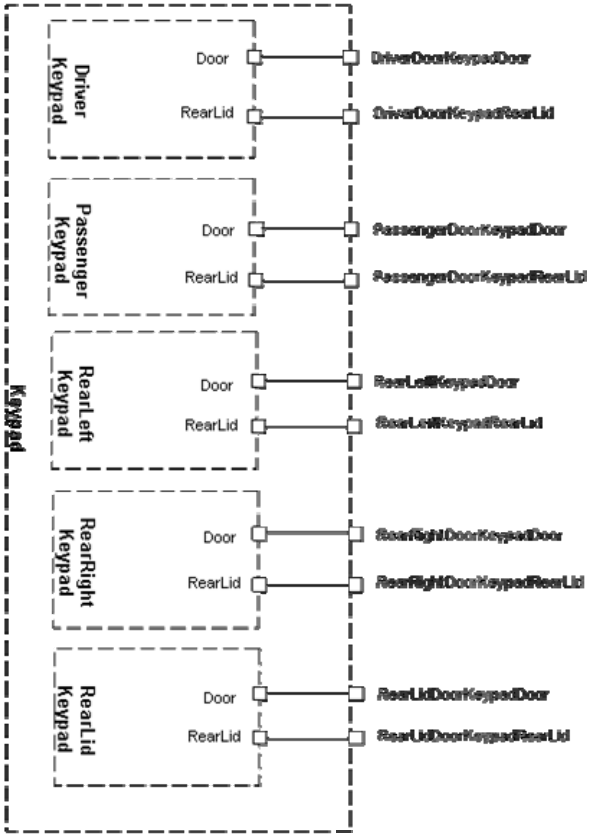


The above figure shows the further decomposition of doors.

Keypad

Out of the keypad the request to activate the CL-functionality for doors is generated as well as the request to seperately activate rear lid.

Keypad is instantiated in front doors, rear doors and rear lid (see figure below).



GloveBox / Tankflap
Other blocks like Glovebox and TankFlap provide status information and (partially) receive locking commands. There is no use case for engine-hood.

Intra-Domain-Interfaces
Through Intra-Domain-Interfaces intra-domain information is provided to other body and comfort systems (please note that scope of this section is only central locking).

Examples:
- Comfort open/close functionality is requested from windows as well as sunroof (see for example separate documentation for WindowControl).
- Visual feedback is requested from Blink Master (see separate documentation for ExteriorLight).

Inter-Domain-Interfaces
Through Inter-Domain-Interfaces inter-domain status information is exchanged.

ProfileManager
It is supposed only a single source coordinates profile Ids from certain personalization sources (remote key, keyless etc.). The ProfileManager is NOT in scope of this section.

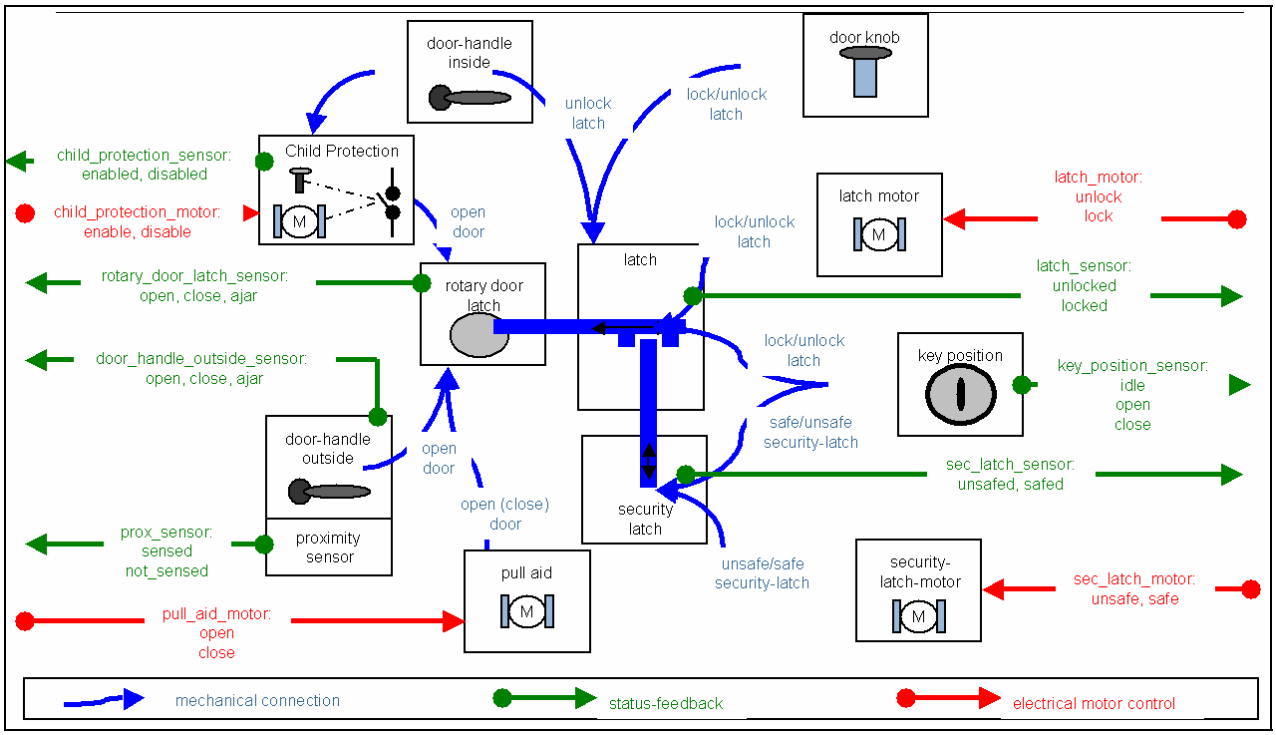
4.5.3 Furhter Issues concerning Central Locking

Personalisation
It is supposed that the Central Locking Master usually is personalizable.

Known use cases for personalization of central locking functionality are for example:

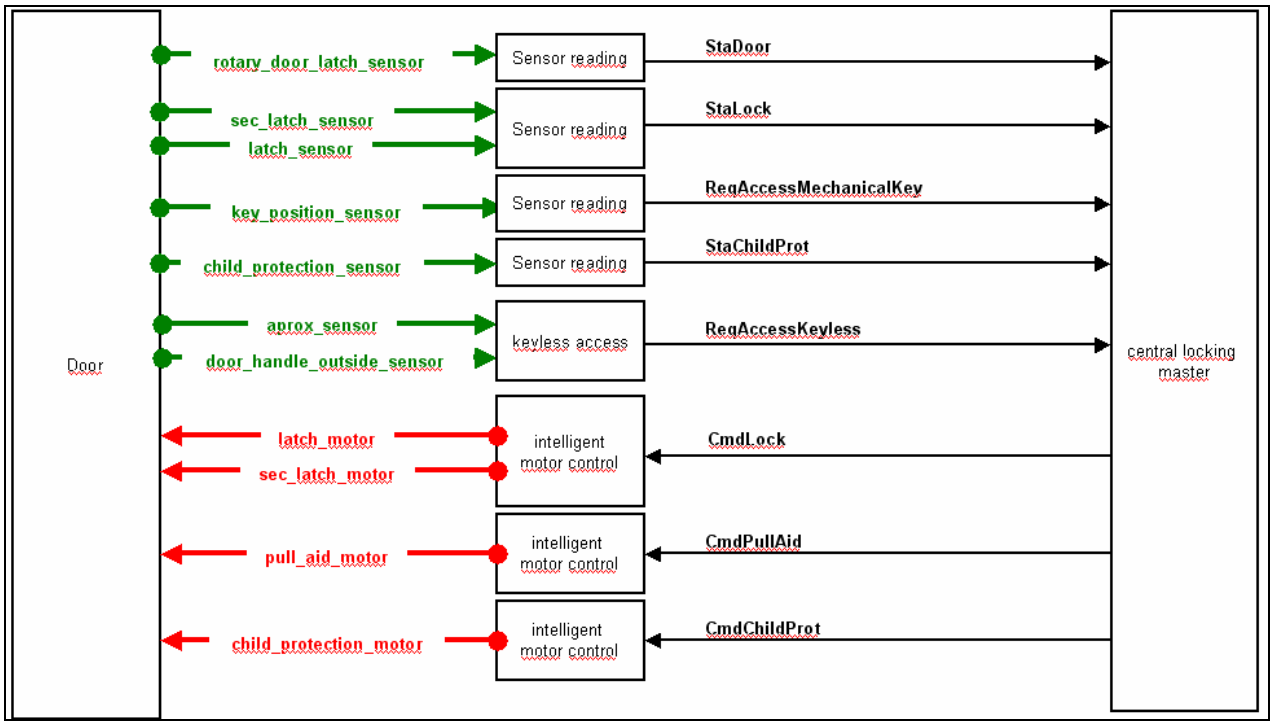
- “Speedlock”: lock all doors at (or above) a certain vehicle speed.
- Selective door opening: open only driver door or open all doors.
- Enable/disable certain central locking functionality (e.g. visual feedback).

Mechanical decomposition door



The figure above is intended for clarification of further mechanical relations.

Mechanical actions vs. standardized interfaces (door only)



The figure demonstrates the relation of mechanical actions at the door with standardized interface.

4.5.4 Known Defects

Diagnostics: Diagnostics Ports are not part of this explanatory description. Nevertheless, diagnosic interfaces/ports are defined (see Master Table).

4.6 Exterior Light

The `ExteriorLight` software component group controls the exterior light functionality of a car. `ExteriorLight` receives driver wishes, senses the environment and controls the behavior of the exterior light sources of a car.

For the current release only the according interfaces to other functionalities are specified.

5 Additional Information (Optional)

This chapter shall contain information that doesn't fit into the previous chapters.
Proper sub-titles needed.

This design standard is the AUTOSAR-architecture for Body domain functionalities. It provides a decomposition into SW-Cs and a list of standardized interfaces related to them.
Each decomposition is limited in granularity to sensor components, adapter components, the core functionality and actuator components. The intent is for the decomposition to get not to atomic SWCs but to "purchasable" SW-Cs. These SW-Cs will be obtained as a unit so that all the internal (and hence not AUTOSAR standardized) interfaces are controlled by a single vendor, even if there are SW-Cs within the bought unit that reside on different ECUs. All interfaces between SW-Cs from different vendors should be standardized.
In addition, all ports are described, showing their AUTOSAR data qualities. Invalidation (where needed) is defined as in-band invalidation. Init values are specified where appropriate e.g. "off", "idle", "undefined", "unknown" … Where standardization is inappropriate, recommendations may be provided.