Delta-oriented Software Product Line Test Models –

The Body Comfort System Case Study

Informatik-Bericht Nr. 2012 – 07

Sascha Lity[1], Remo Lachmann[1], Malte Lochau[2], and Ina Schaefer[1]

[1]Institute for Software Engineering and Automotive Informatics, [2]Real-Time Systems Lab (TU Darmstadt)

April 26, 2017

Acknowledgement

The author Sascha Lity is supported by the DFG (German Research Foundation) under the Priority Programme SPP1593: Design For Future – Managed Software Evolution.

Changelog

Version	Date	Author	Change Description
1.0	17.04.2013	Sascha Lity	Version 1.0 finalized
1.1	23.07.2014	Sascha Lity	Typo in Figure 4.24 fixed
1.2	17.06.2015	Remo Lachmann	Updated architecture graphs for product P2, P12 and P13, added architecture delta DLEDEMH, updated and reformatted architecture deltas. Added MSC65 for full coverage.
1.3	13.04.2017	Remo Lachmann	Added BCS requirements and system test cases, updated author affiliations.
1.4	21.04.2017	Sascha Lity	Change document template.

Abstract

In this Technical Report, we present a complete description of a delta-oriented software product line test model for component and integration testing of a Body Comfort System SPL case study from the automotive domain. The test model comprises the component interface specifications and their variants and the component integration in our architecture test models, as well as the interaction test scenarios for integration testing by means of message sequence charts. As test models for component testing we use state machines.

Contents

List of Tables

List of Figures

Listings

List of Abbreviations

AL	Automatic Locking
AM	Architecture Model
AS	Alarm System
ATM	Architecture Test Model
AutoPW	Automatic Power Window
BCS	Body Comfort System
CAN	Controller Area Network
CAS	Control Alarm System
CAP	Control Automatic Power Window
CLS	Central Locking System
E/E	Electric/Electronic
ECU	Electronic Control Unit
FM	Feature Model
HMI	Human Machine Interface
IM	Interior Monitoring
ITM	Integration Test Model
LED	Status LED
LED AS	LED Alarm System
LED CLS	LED Central Locking System
LED EM	LED Exterior Mirror
LED FP	LED Finger Protection
LED Heat	LED Heatable
LED PW	LED Power Window
ManPW	Manual Power Window
MSC	Message Sequence Chart
PW	Power Window
RCK	Remote Control Key
SF	Safety Function
SM	State Machine
SPL	Software Product Line

1 Introduction

Embedded software systems have become an integral part of modern automotive system development for implementing complex control and regulation tasks. A sample E/E architecture of an automotive software system is illustrated in Fig 1.1 constituting a simplified *Body Comfort System (BCS)* comprising functionality for controlling the *door systems*, a *human machine interface (HMI)*, an *alarm system*, and the *central locking system*. The component-based

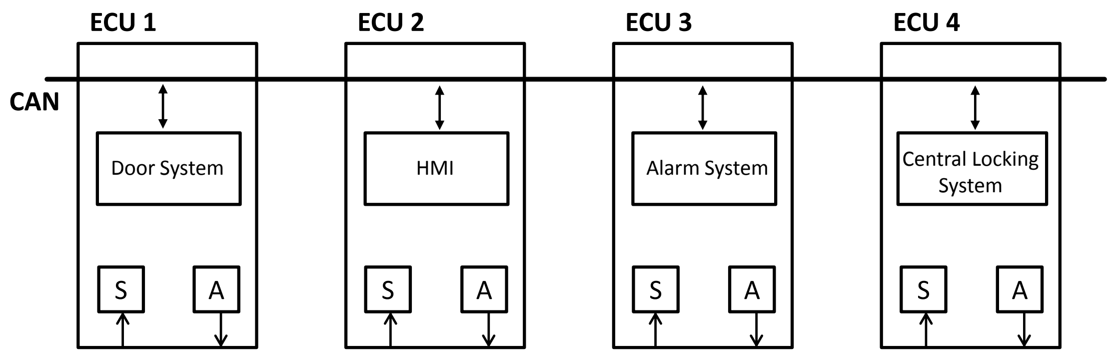

Figure 1.1.: E/E Architecture of the BCS Case Study.

E/E architecture consists of a collection of distributed electronic control units (ECU) each dedicated to run the software functions of a particular sub system and communicating with each other via a CAN bus. The corresponding software components, therefore, implement reactive control tasks interacting with each other and with the environment via input signals provided by sensors (S) and output signals emitted to actuators (A). In [6], we provide a detailed description of a sample model-based software development and quality assurance methodology for automotive systems by means of the BCS case study.

Model-based testing offers a comprehensive approach for efficient, yet reliable quality assurance especially for software developed for safety critical systems as apparent in the automotive domain. Based on a test model, i.e., a formal specification of the intended behavior of the (software) system under test, model-based testing aims at the automated derivation, execution and evaluation of test suites satisfying some adequacy criteria, e.g., test model coverage [10]. Model-based testing is applicable at different architectural levels focusing on component testing and integration testing.

Apart from the reactivity and inherent complexity of automotive (software) systems, their development and efficient quality assurance is further obstructed by their ever-growing diversity. An automotive system, therefore, comprises a family of similar system variants rather than a singleton, monolithic system implementation. Software product line (SPL) engineering provides a promising approach for developing and maintaining variant-rich software systems in a concise way [2, 8]. During domain engineering, domain feature models are usually used to specify the commonality and variability among the different members of a product family in terms of their supported features [3]. A mapping of those features as configuration parameters onto reusable engineering artifacts within the solution space allows for an automated configuration and efficient assembling of product variant implementations during application engineering.

In [11, 7], the BCS case study is re-engineered as an SPL. Thereupon, we developed a framework for model-based testing approaches applying the reuse principles propagated by SPL engineering also to test artifact design and test suite execution [1, 5, 4]. The approach aims at efficient, redundancy-reduced testing strategies for SPL, yet ensuring a reliable test coverage of every product variant. The approaches are based on a delta-oriented architecture-based SPL test model including

- a variable architecture model by means of a component-connector specification,
- variable component test models by means of delta-oriented state machines and
- variable integration test models by means of collections of message sequence charts.

In this technical report, we provide a complete coverage of this BCS SPL test model collection.

The remainder of the report is organized as follows. In Sect. 2, an overview of the Body Comfort System SPL case study is given. We document the variable component interface specifications and the delta-oriented architecture models in Sect. 3. Section 4 describes the delta-oriented component state machine test models. We define the integration test models in Sect. 5. Section 6 documents the aggregation of the different models to the delta-oriented architecture test models and Sect. 7 concludes the report. In the appendix, we present the 150% Model of the BCS SPL as well as the complete DELTARX specification. In addition, the system requirements and the corresponding system test cases for the original BCS case study are documented.

2 Overview of BCS SPL Case Study

The BCS case study originally developed by Müller et al. (cf. [6]) in cooperation with industrial partners from the automotive sector comprises the following functionality.

- Electric *Power Window* with *Finger Protection*
- Electric and heatable *Exterior Mirror*
- *Human Machine Interface* with *Status LEDs*
- *Alarm System* with *Interior Monitoring*
- *Central Locking System* with *Automatic Locking*
- *Remote Control Key* with *Safety Function*, *Alarm System Control*, and *Power Window Control*

By *Automatic Locking*, the *Central locking System* provides the automated locking of the doors when the car is driving. By *Safety Function*, the *Remote Control Key* provides the automated locking of the car after a specific timeout, i.e., the car is unintentionally unlocked.

The BCS was enhanced to a software product line by Oster et al. [11, 7] by decomposition into variable parts, i.e., by making the *Alarm System*, the *Central Locking System*, and the *Remote Control Key* optional functions. For the specification of the variability of the BCS SPL case study, a feature model is defined as shown in Fig. 2.1. Here, each system function is represented by a corresponding feature.

The feature model comprises 27 *features*, five *require* constraints, e.g., the selection of feature *LED Heatable* requires the selection of *Heatable*, and one *exclude* constraint, e.g., the features *Control Automatic Power Window* and *Manual Power Window* cannot be selected for the same product variant. In contrast to the original version of the BCS case study, we now have two alternatives of the *Power Window*. The *Manual Power Window* moves up/down when pressing and holding the button for the window movement and the *Automatic Power Window* moves up/down when pressing the button for the window movement once . Considering all possible feature combinations, the BCS SPL comprises 11.616 valid product variants.

Based on the results of MoSo-PoLiTe [11, 7] where the BCS SPL case study was evaluated as well for combinatorial pairwise subset testing, we focus in this documentation on the therein identified representative subset of 17 product variants. The original 150% state machine test model developed for the MoSo-PoLiTe evaluation, is shown in the appendix A.1.

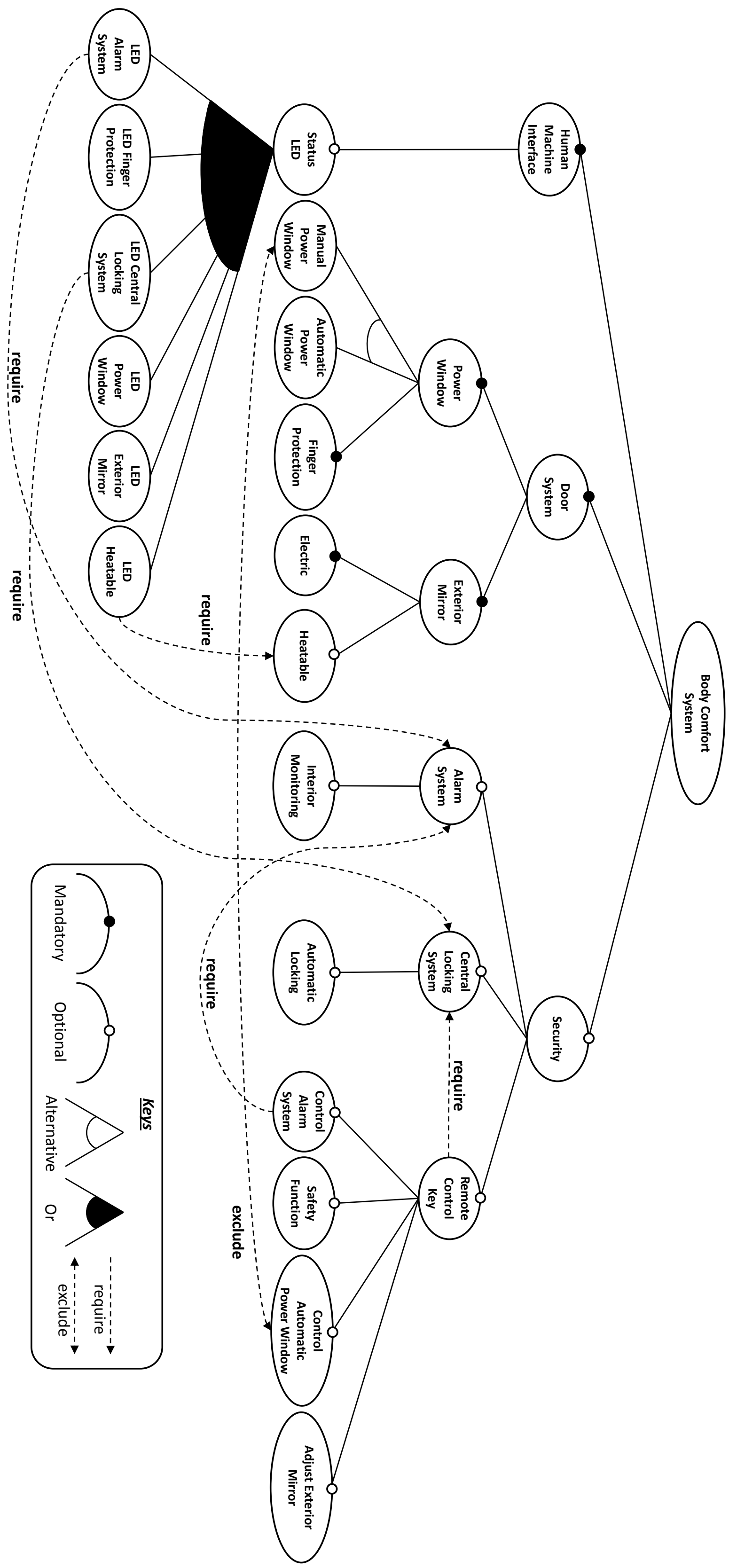

Figure 2.1.: Feature Model of the Body Comfort System SPL Case Study

The component state machine test models described in this report are slightly different to the original 150% model. The documented artifacts for the different types of test models are further valid and sufficient to obtain the test model variants for the remaining product variants. In previous work [5, 4], we added another valid product to the representative set. We use this variant as the core product comprising the most commonalities among the various product variants. The corresponding feature configurations of the 18 products under test are listed in the following Tab. 2.1 and Tab. 2.2. By $P0$, we refer to the core product. Please note that the tables solely list the variable features of the BCS SPL feature model.

	$P0$	$P1$	$P2$	$P3$	$P4$	$P5$	$P6$	$P7$	$P8$
Security		x	x	x	x	x	x		
Status LED		x		x		x	x		x
LED Central Locking System				x			x		
LED Power Window				x		x	x		x
LED Heatable				x		x	x		x
LED Exterior Mirror				x		x	x		x
LED Alarm System				x		x	x		
LED Finger Protection		x				x	x		x
Manual Power Window	x			x		x	x		
Automatic Power Window		x	x		x			x	x
Heatable		x		x		x	x		x
Central Locking System		x	x	x	x		x		
Remote Control Key		x	x	x	x				
Alarm System		x	x	x	x	x	x		
Automatic Locking			x		x		x		
Control Alarm System		x	x	x					
Safety Function			x		x				
Control Automatic PW			x		x				
Interior Monitoring		x			x	x	x		
#Features	1	10	9	13	9	11	14	1	7

Table 2.1.: Overview of Product Configurations $P0$-$P8$ with Variable Features

As already mentioned, each feature corresponds to a specific system function. Thus, features are mapped to specific sub systems (components) and/or enhancements of the core functionality. In the next section, we describe the variable component interface specifications and their combinations in the delta-oriented architecture models for the 18 product variants. We use those architecture models as a basis for the definition of the different types of test models for the component as well as for integration testing presented in the remainder of the report.

	*P*9	*P*10	*P*11	*P*12	*P*13	*P*14	*P*15	*P*16	*P*17
Security	x		x	x	x	x	x	x	
Status LED	x	x	x		x			x	
LED Central Locking System	x		x		x			x	
LED Power Window	x		x						
LED Heatable	x				x			x	
LED Exterior Mirror	x	x						x	
LED Alarm System	x		x					x	
LED Finger Protection	x	x	x		x			x	
Manual Power Window		x	x		x			x	x
Automatic Power Window	x			x		x	x		
Heatable	x			x	x	x	x	x	x
Central Locking System	x		x	x	x	x	x	x	
Remote Control Key	x		x	x	x	x	x	x	
Alarm System	x		x			x	x	x	
Automatic Locking			x	x	x	x	x	x	
Control Alarm System	x		x			x	x	x	
Safety Function	x		x	x	x	x		x	
Control Automatic PW	x			x		x	x		
Interior Monitoring	x		x			x	x	x	
#Features	17	4	14	8	11	11	10	16	2

Table 2.2.: Overview of Product Configurations *P*9-*P*17 with Variable Features

3 BCS Architecture Model

In this section, we describe the component interface specifications and their variants as well as the different architecture model variants. In addition to the core architecture model, we further list the architecture model deltas transforming the core architecture to a specific variant.

As a basis for the BCS architecture model documentation, Fig. 3.1 gives an overview of all component-connector specifications, where feature annotations are omitted for a better readability. Please note that the illustrated architecture is not a valid architecture model for the BCS SPL case study. Each component represents a specific system function defined by the corresponding feature, whereas internal connectors represent interactions between components and input/output connectors constitute the communication with the system environment. In the following, we describe in detail the various component variants and their valid combinations in architecture models for the representative subset of product variants ($P0$-$P17$).

3.1. Component Interface Specifications

Manual Power Window The standard component for the feature *Manual Power Window* is depicted in Fig. 3.2, controlling the movement of the window. The interface is defined by the following input and output signals.
Input Signals:

- *pw_but_dn* controls the downwards movement of the window
- *pw_but_up* controls the upwards movement of the window
- *pw_pos_up* indicates the upper window position
- *pw_pos_dn* indicates the lower window position
- *fp_on* stops and disables the upwards movement of the window
- *fp_off* re-enables the upwards movement

Output Signals:

- *pw_mv_up* represents the upwards movement of the window
- *pw_mv_dn* represents the downwards movement of the window

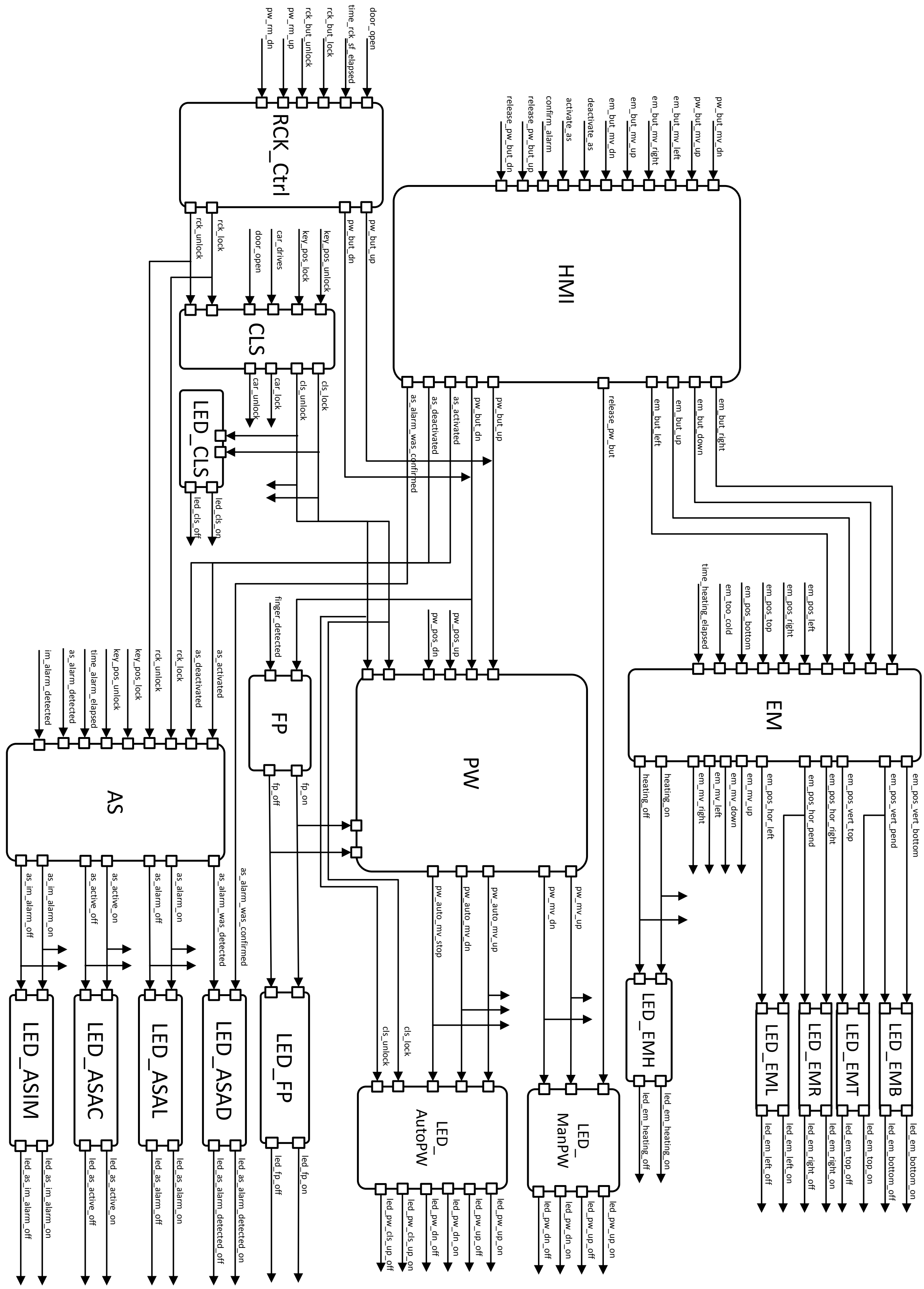

Figure 3.1.: Overview of the Variable Architecture Models

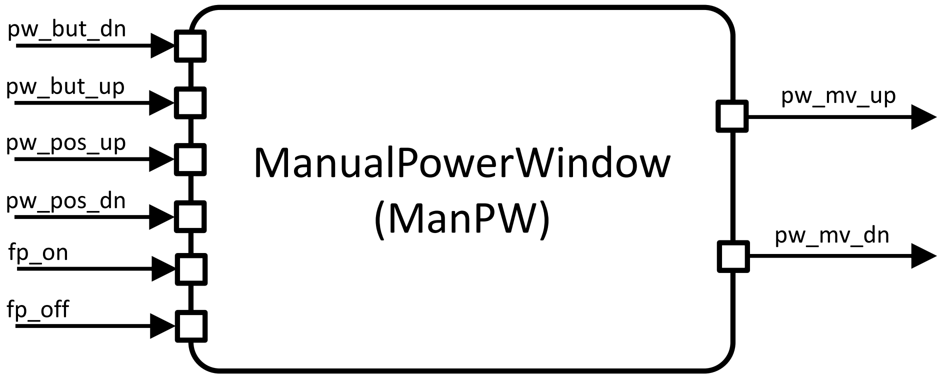

Figure 3.2.: Interface Specification of the Standard ManualPowerWindow Component

Based on the feature *Central Locking System*, the standard specification is extended such that the window movement is stopped and disabled by the locking of the car and re-enabled by unlocking the car. The new interface is shown in Fig. 3.3, defining further input signals.

- *cls_lock* stops and disables the window movement
- *cls_unlock* re-enables the window movement

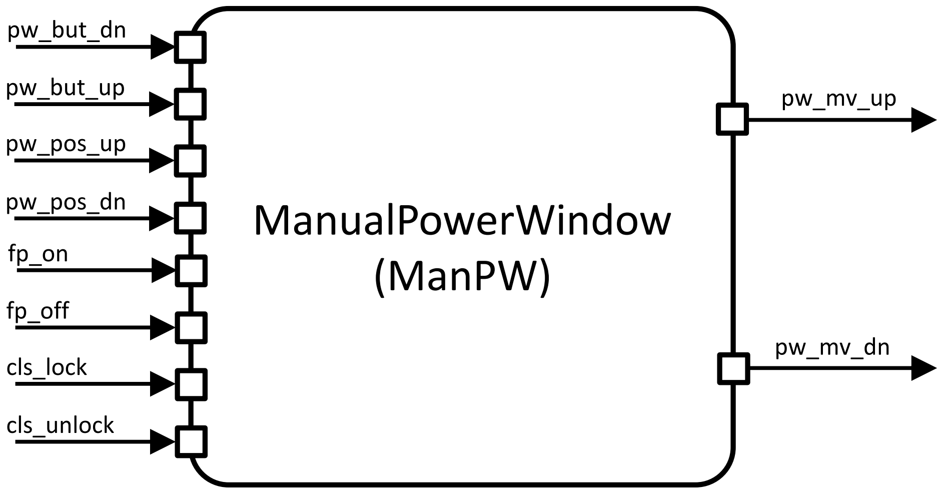

Figure 3.3.: Interface Specification of the ManualPowerWindow Component Variant with Central Locking System Feature

Automatic Power Window The standard component for the feature *Automatic Power Window* is depicted in Fig. 3.4, controlling the automated movement of the window. The interface is defined by the following input and output signals.
Input Signals:

- *pw_but_dn* controls the automated downwards movement of the window

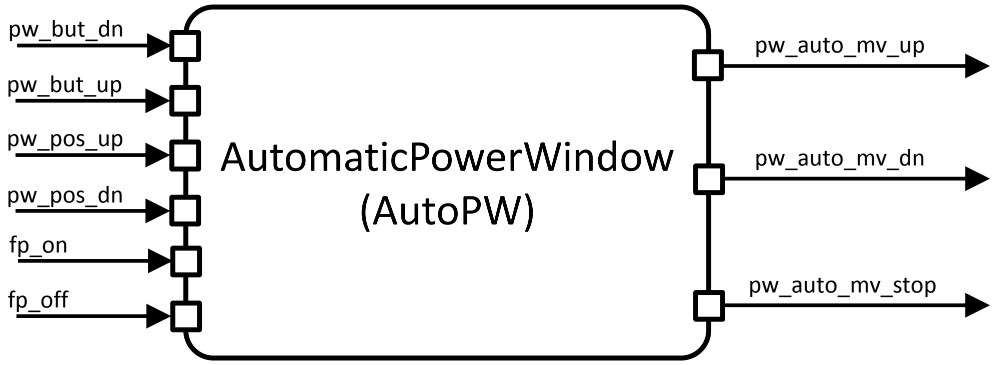

Figure 3.4.: Interface Specification of the Standard AutomaticPowerWindow Component

- *pw_but_up* controls the automated upwards movement of the window
- *pw_pos_up* indicates the upper window position
- *pw_pos_dn* indicates the lower window position
- *fp_on* stops and disables the upwards movement of the window
- *fp_off* re-enables the upwards movement

Output Signals:

- *pw_auto_mv_up* represents the automated upwards movement of the window
- *pw_auto_mv_dn* represents the automated downwards movement of the window
- *pw_auto_mv_stop* stops the automated movement of the window

Based on the feature *Central Locking System*, the standard specification is extended such that the window movement is stopped and disabled by the locking of the car and re-enabled by unlocking the car. The new interface is shown in Fig. 3.5, defining further input signals.

- *cls_lock* stops and disables the automated window movement
- *cls_unlock* re-enables the automated window movement

Finger Protection The standard component for the feature *Finger Protection* is depicted in Fig. 3.6, controlling the disabling/re-enabling of the window movement based on a clamped finger. There are no further component variants. The interface is defined by the following input and output signals.
Input Signals:

- *finger_detected* represents the detection of a clamped finger

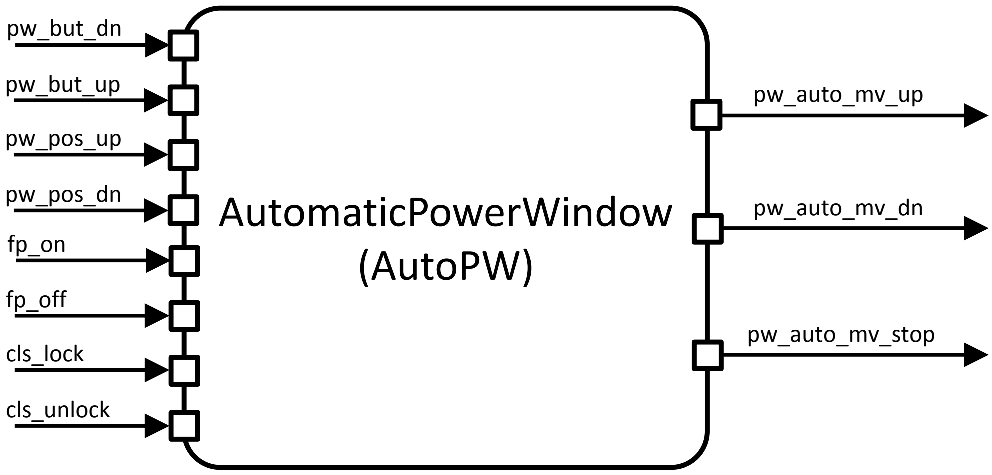

Figure 3.5.: Interface Specification of the AutomaticPowerWindow Component Variant with Central Locking System Feature

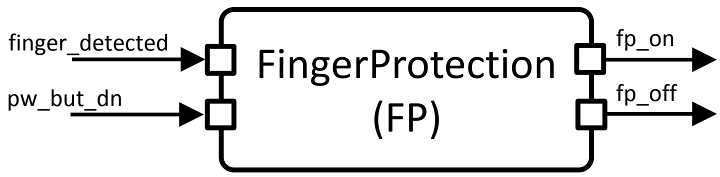

Figure 3.6.: Interface Specification of the FingerProtection Component

- *pw_but_dn* releases the finger based on the downwards movement of the window

Output Signals:

- *fp_on* represents the activation of the finger protection
- *fp_off* represents the deactivation of the finger protection

Exterior Mirror The standard component for the feature *Exterior Mirror* is shown in Fig. 3.7, controlling the mirror movement. The standard interface is defined by the following input and output signals.

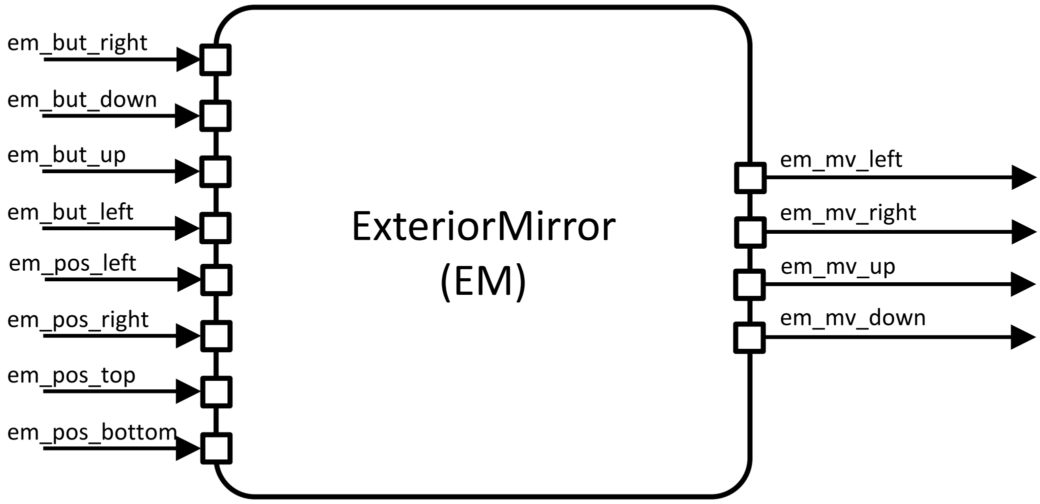

Figure 3.7.: Interface Specification of the Standard ExteriorMirror Component

Input Signals:

- *em_but_right* controls the rightwards movement of the exterior mirror
- *em_but_down* controls the downwards movement of the exterior mirror
- *em_but_up* controls the upwards movement of the exterior mirror
- *em_but_left* controls the leftwards movement of the exterior mirror
- *em_pos_left* indicates the left-most mirror position
- *em_pos_right* indicates the right-most mirror position
- *em_pos_top* indicates the upper mirror position
- *em_pos_bottom* indicates the lower mirror position

Output Signals:

- *em_mv_left* represents the leftwards movement of the exterior mirror

- *em_mv_right* represents the rightwards movement of the exterior mirror
- *em_mv_up* represents the upwards movement of the exterior mirror
- *em_mv_down* represents the downwards movement of the exterior mirror

Based on the feature *Heatable*, the mirror is heatable. The new interface is depicted in Fig. 3.8, defining further input and output signals.

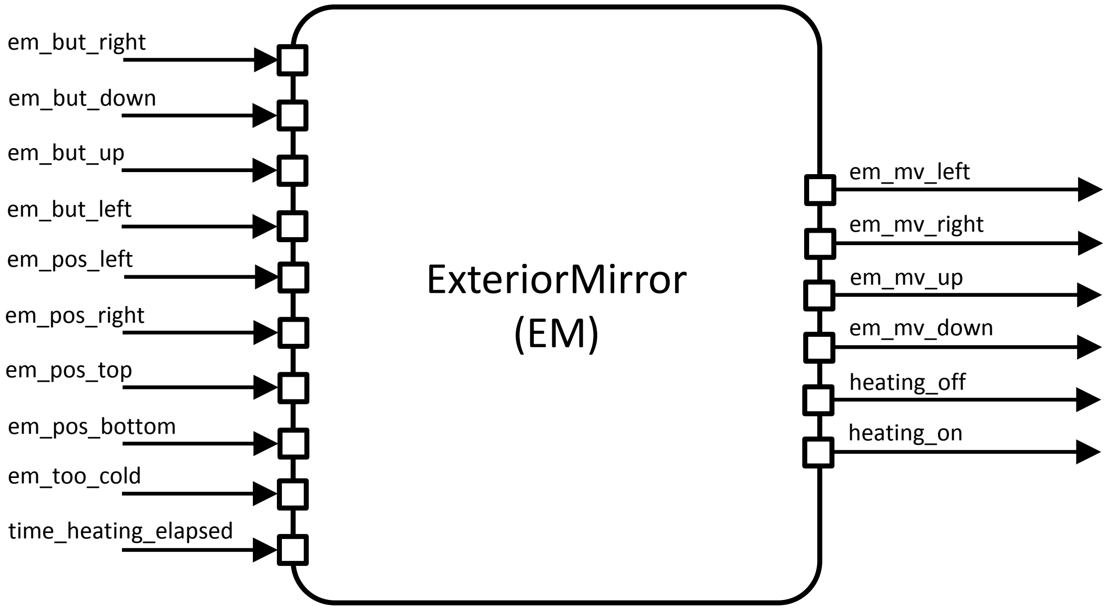

Figure 3.8.: Interface Specification of the ExteriorMirror Component Variant with Heatable Feature

Input Signals:

- *em_too_cold* controls the activation of the mirror heater
- *time_heating_elapsed* controls the deactivation of the mirror heater based on the elapsed heating time

Output Signals:

- *heating_off* stops the mirror heater
- *heating_on* starts the mirror heater

Based on the feature *LED Exterior Mirror*, the standard specification is extended such that the mirror provides the information about its current position to the corresponding LEDs. The new interface is shown in Fig. 3.9, defining further output signals.

- *em_pos_vert_pend* represents the mirror movement between the upper and lower mirror position

- *em_pos_vert_bottom* represents the lower mirror position
- *em_pos_vert_top* represents the upper mirror position
- *em_pos_hor_pend* represents the mirror movement between the left-most and right-most mirror position
- *em_pos_hor_right* represents the right-most mirror position
- *em_pos_hor_left* represents the left-most mirror position

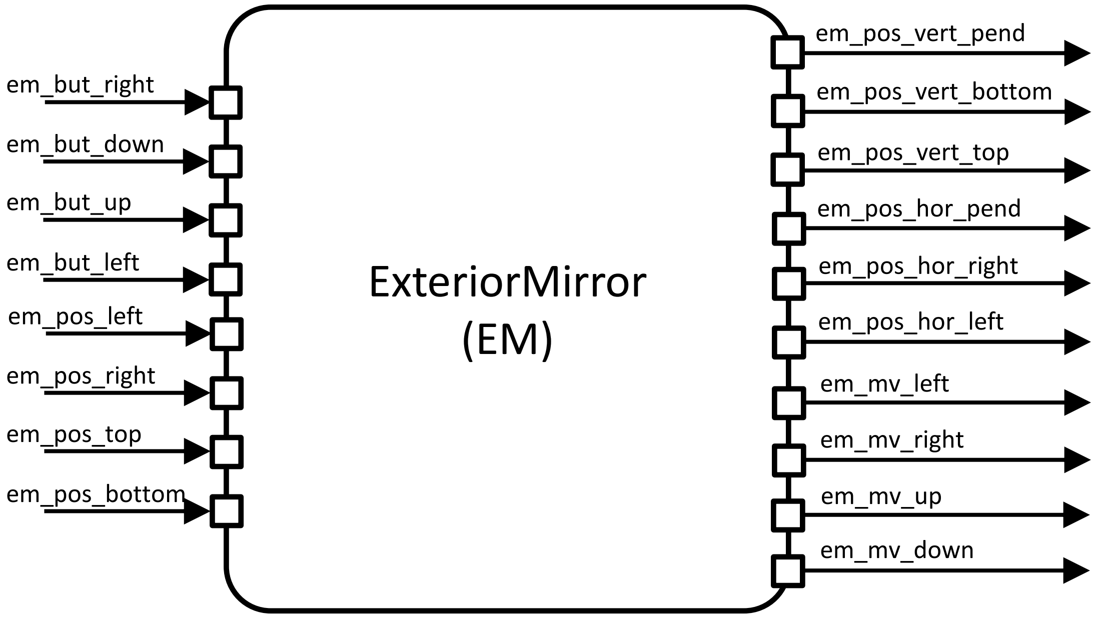

Figure 3.9.: Interface Specification of the ExteriorMirror Component Variant with LED Exterior Mirror Feature

Based on the potential feature combination *LED Exterior Mirror* and *Heatable*, another component variant is defined as depicted in Fig. 3.10, combining the signals of the standard specification as well as the signals of the two previously defined variants.

Alarm System The standard component for the feature *Alarm System* is shown in Fig. 3.11, controlling the activation/deactivation of the alarm system as well as the triggering of the alarm. The interface is defined by the following input and output signals.
Input Signals:

- *key_pos_lock* enables the alarm monitoring
- *key_pos_unlock* disables the alarm monitoring
- *as_alarm_detected* triggers the alarm
- *time_alarm_elapsed* represents the timeout after which the alarm signal stops

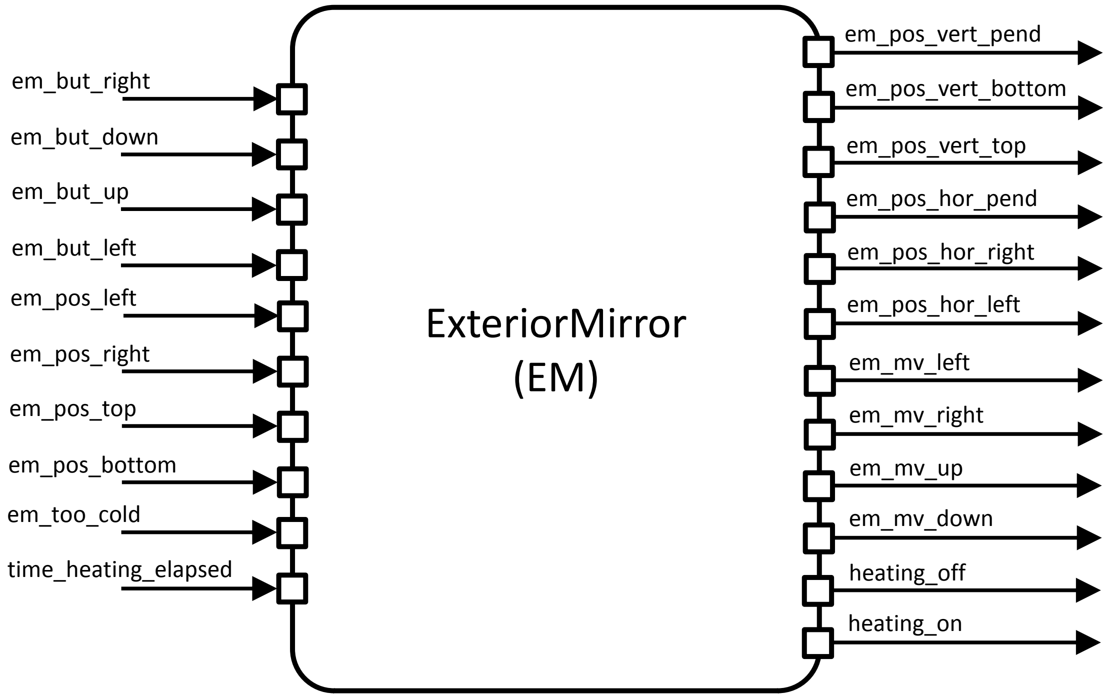

Figure 3.10.: Interface Specification of the ExteriorMirror Component Variant with LED Exterior Mirror and Heatable Features

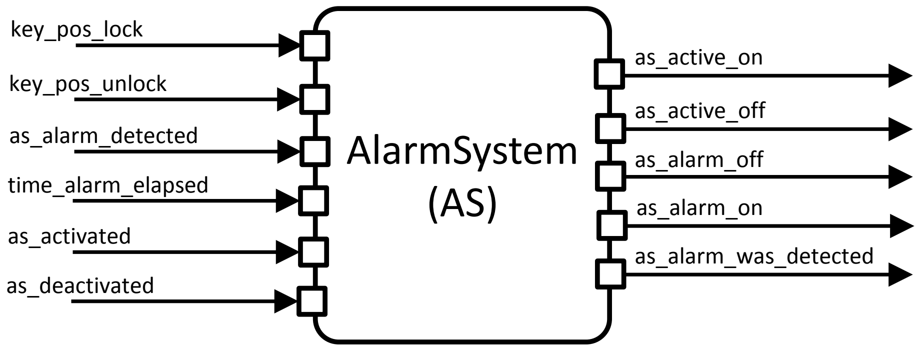

Figure 3.11.: Interface Specification of the Standard AlarmSystem Component

- *as_activated* controls the activation of the alarm system
- *as_deactivated* controls the deactivation of the alarm system

Output Signals:

- *as_active_on* activates the alarm monitoring
- *as_active_off* deactivates the alarm monitoring
- *as_alarm_off* stops the alarm signal
- *as_alarm_on* starts the alarm signal
- *as_alarm_was_detected* represents a silent alarm after the alarm time elapsed

Based on the feature *Control Alarm System*, the standard specification is extended such that the alarm monitoring of the alarm system is enabled/disabled by the remote key. The new interface specification is depicted in Fig.3.12, defining new input signals.

- *rck_lock* enables the alarm monitoring
- *rck_unlock* disables the alarm monitoring

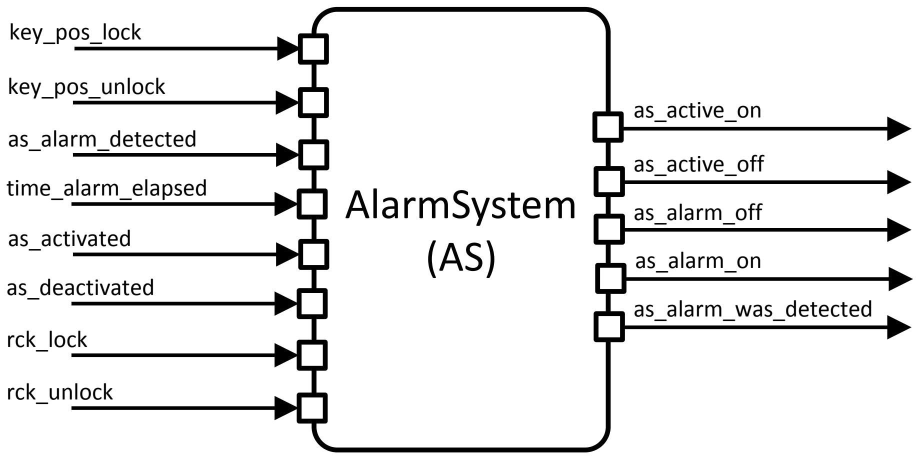

Figure 3.12.: Interface Specification of the AlarmSystem Component Variant with Control Alarm System Feature

Based on the feature *Interior Monitoring*, the standard specification is extended such that the alarm is triggered when the monitoring system detects something inside the car. The interface variant is shown in Fig. 3.13, defining further input and output signals.
Input Signals:

- *im_alarm_detected* triggers the alarm based on the interior monitoring system

Output Signals:

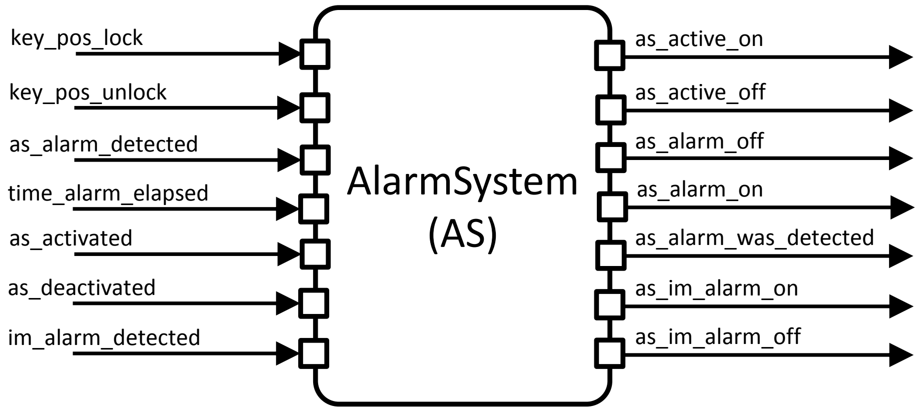

Figure 3.13.: Interface Specification of the AlarmSystem Component Variant with Interior Monitoring Feature

- *as_im_alarm_on* starts the interior alarm signal
- *as_im_alarm_off* stops the interior alarm signal

Based on the potential feature combination *Control Alarm System* and *Interior Monitoring*, another component variant is defined as shown in Fig. 3.14, combining the signals of the standard specification as well as the signals of the two previously defined variants.

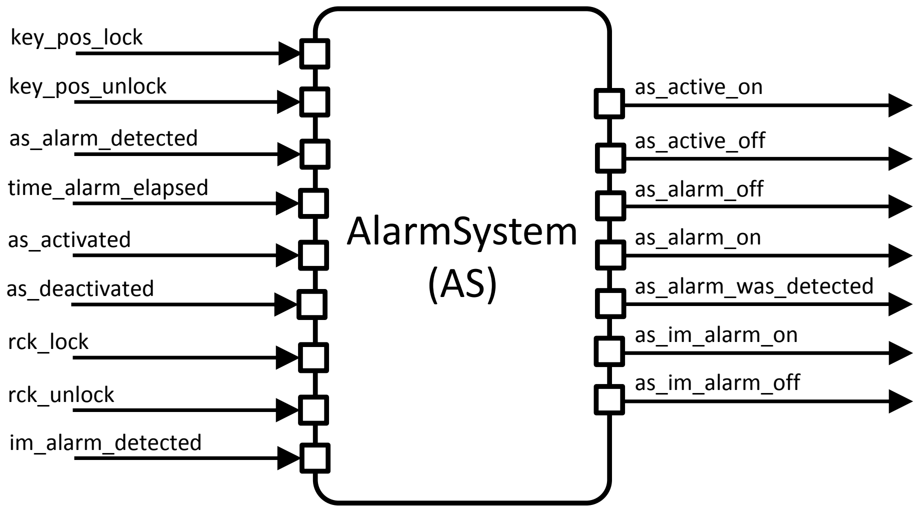

Figure 3.14.: Interface Specification of the AlarmSystem Component Variant with Control Alarm System and Interior Monitoring Features

Remote Control Key Controller The standard component for the feature *Remote Control Key* is depicted in Fig. 3.15, enabling the remote control of the central locking system. The standard interface is defined by the following input and output signals.

Figure 3.15.: Interface Specification of the Standard RemoteControlKeyController Component

Input Signals:

- *rck_but_lock* represents the remote signal for locking the car
- *rck_but_unlock* represents the remote signal for unlocking the car

Output Signals:

- *rck_lock* controls the locking of the car
- *rck_unlock* controls the unlocking of the car

Based on the feature *Control Automatic Power Window*, the standard specification is extended such that the remote key enables the remote control of the automated window movement. The new interface is shown in Fig. 3.16, defining further input and output signals.

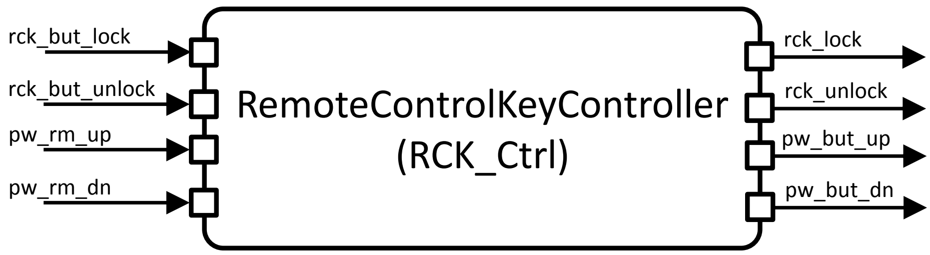

Figure 3.16.: Interface Specification of the RemoteControlKeyController Component Variant with Control Automatic Power Window Feature

Input Signals:

- *pw_rm_up* represents the remote signal for the upwards movement of the window
- *pw_rm_dn* represents the remote signal for the downwards movement of the window

Output Signals:

- *pw_but_up* controls the upwards movement of the window
- *pw_but_dn* controls the downwards movement of the window

Based on the feature *Safety Function*, the standard specification is extended such that the car is locked again (timeout occurs) if the car was unintentionally unlocked by the remote signal. The new interface is depicted in Fig. 3.17, defining further input signals.

- *door_open* represents an open door, i.e., the car was intentionally unlocked by the remote signal
- *time_rck_sf_elapsed* represents the timeout after which the car is locked again

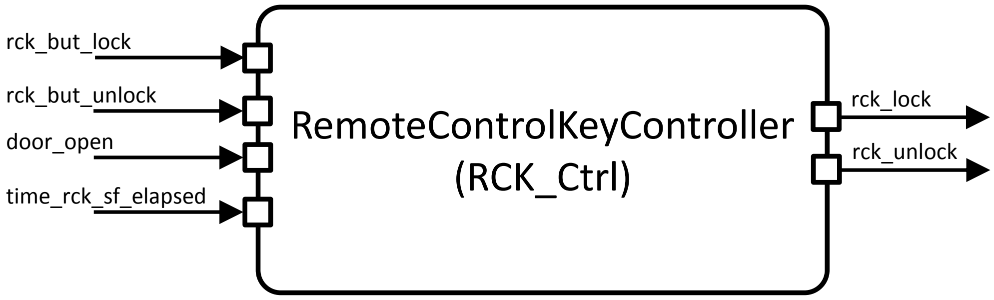

Figure 3.17.: Interface Specification of the RemoteControlKeyController Component Variant with Safety Function Feature

Based on the potential feature combination *Control Automatic Power Window* and *Safety Function*, another component variant is defined as shown in Fig. 3.18, combining the signals of the standard specification as well as the signals of the two previously defined variants.

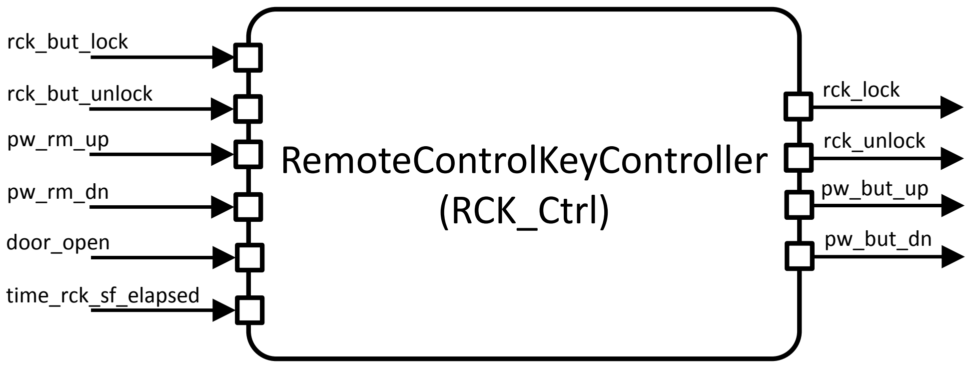

Figure 3.18.: Interface Specification of RemoteControlKeyController Component Variant with Control Automatic Power Window and Safety Function Features

Central Locking System The standard component for the feature *Central Locking System* is depicted in Fig. 3.19, controlling the locking/unlocking of the car. The standard interface is defined by the following input and output signals.
Input Signals:

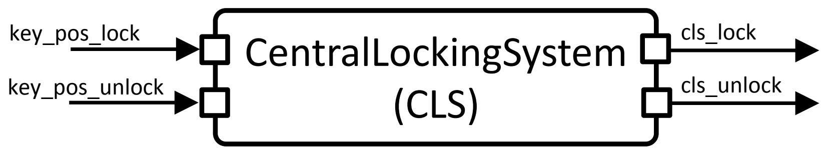

Figure 3.19.: Interface Specification of Standard CentralLockingSystem Component

- *key_pos_lock* controls the locking of the car
- *key_pos_unlock* controls the unlocking of the car

Output Signals:

- *cls_lock* locks the car and disables the window movement
- *cls_unlock* unlocks the car and re-enables the window movement

Based on the feature *Remote Control Key*, the standard specification is extended such that the locking/unlocking is further controlled by the remote key. The new interface is shown in Fig. 3.20, defining further input signals.

- *rck_lock* controls the locking of the car (remote)
- *rck_unlock* controls the unlocking of the car (remote)

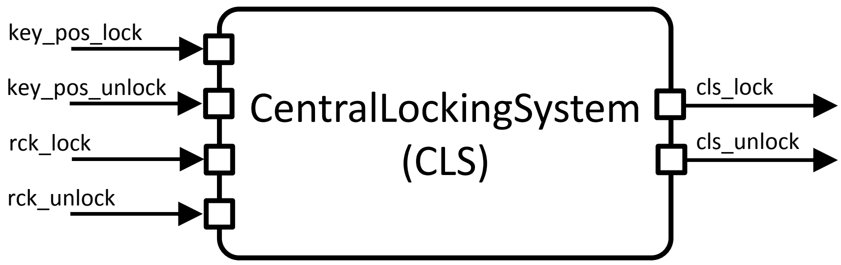

Figure 3.20.: Interface Specification of the CentralLockingSystem Component Variant with Remote Control Key Feature

Based on the feature *Automated Locking*, the standard specification is extended such that the doors are locked when the car is driving. The new interface is depicted in Fig. 3.21, defining further input and output signals.
Input Signals:

- *car_drives* controls the locking of the doors when the car is driving
- *door_open* controls the unlocking of the doors when a door is open

Output Signals:

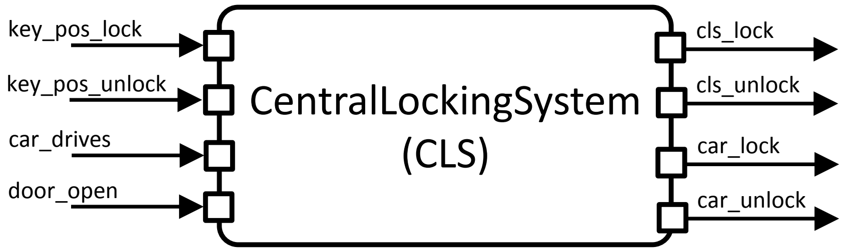

Figure 3.21.: Interface Specification of the CentralLockingSystem Component Variant with Automatic Locking Feature

- *car_lock* locks the doors without disabling the window movement
- *car_unlock* unlocks the doors

Based on the potential feature combination *Remote Control Key* and *Automatic Locking*, another component variant is defined as shown in Fig. 3.22, combining the signals of the standard specification and the signals of the two previously defined variants.

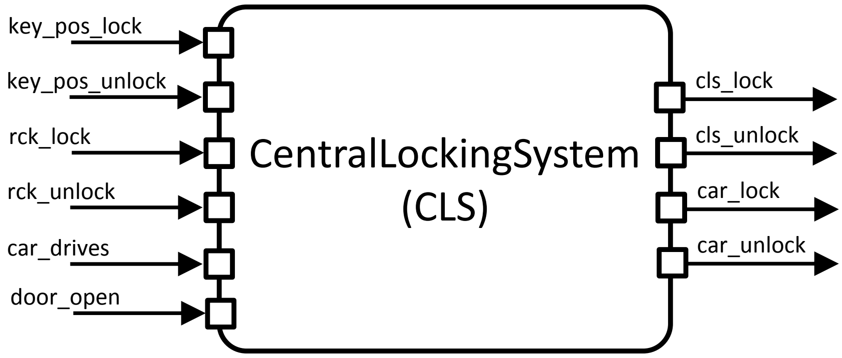

Figure 3.22.: Interface Specification of the CentralLockingSystem Component Variant with Remote Control Key and Automatic Locking Features

Human Machine Interface The standard component for the feature *Human Machine Interface* is depicted in Fig. 3.23, enabling the interaction with a driver. The standard interface is defined by the following input and output signals.
Input Signals:

- *pw_but_mv_dn* represents the signal initiating the downwards movement of the window
- *pw_but_mv_up* represents the signal initiating the upwards movement of the window
- *em_but_mv_left* represents the signal initiating the leftwards movement of the exterior mirror

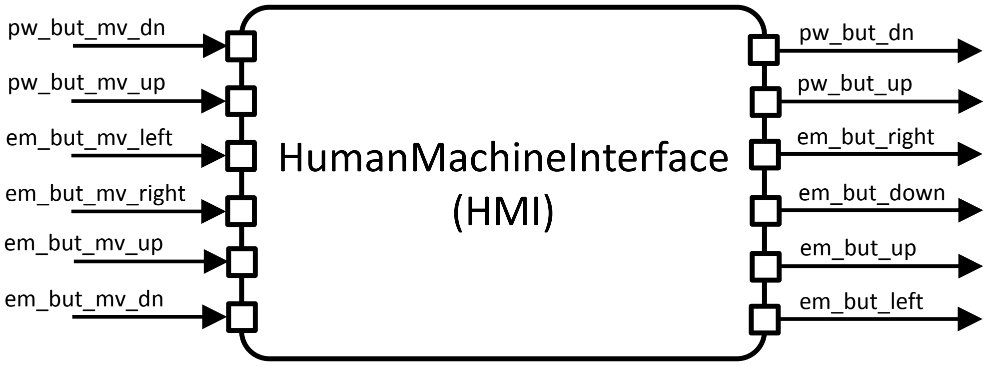

Figure 3.23.: Interface Specification of the Standard HumanMachineInterface Component

- *em_but_mv_right* represents the signal initiating the rightwards movement of the exterior mirror
- *em_but_mv_up* represents the signal initiating the upwards movement of the exterior mirror
- *em_but_mv_dn* represents the signal initiating the downwards movement of the exterior mirror

Output Signals:

- *pw_but_dn* controls the downwards movement of the window
- *pw_but_up* controls the upwards movement of the window
- *em_but_right* controls the rightwards movement of the exterior mirror
- *em_but_left* controls the leftwards movement of the exterior mirror
- *em_but_down* controls the downwards movement of the exterior mirror
- *em_but_up* controls the upwards movement of the exterior mirror

Based on the feature *Alarm System*, the standard specification is extended such that a driver controls (activation/deactivation) the alarm system. The new interface is shown in Fig. 3.24, defining further input and output signals.
Input Signals:

- *deactivate_as* represents the signal initiating the deactivation of the alarm system
- *activate_as* represents the signal initiating the activation of the alarm system

Output Signals:

- *as_activated* activates the alarm system

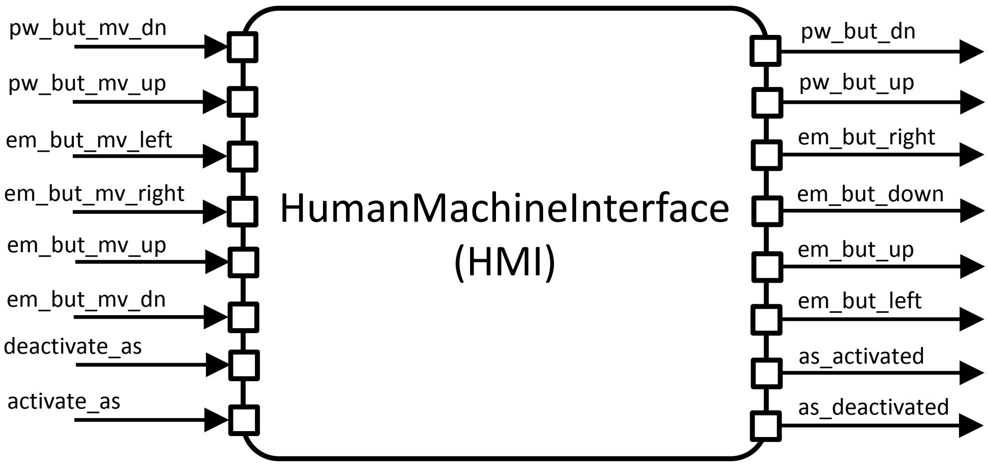

Figure 3.24.: Interface Specification of the HumanMachineInterface Component Variant with Alarm System Feature

- *as_deactivated* deactivates the alarm system

Based on the potential feature combination *Alarm System* and *LED Alarm System*, the standard specification is extended such that a driver is able to confirm the silent alarm. The new interface is depicted in Fig. 3.25, defining, in addition to the signals from the Alarm System variant, further input and output signals.

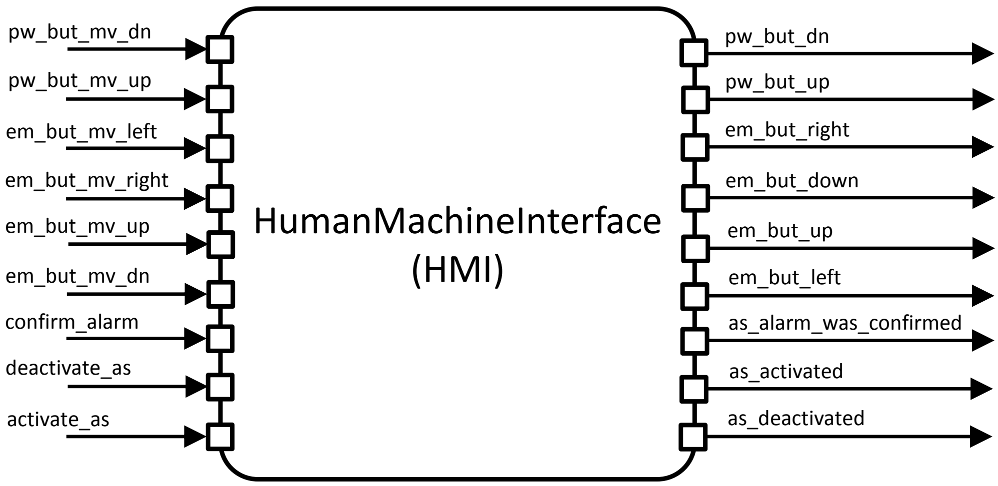

Figure 3.25.: Interface Specification of the HumanMachineInterface Component Variant with Alarm System and LED Alarm System Features

Input Signals:

- *confirm_alarm* represents the signal initiating the confirmation of the alarm

Output Signals:

- *as_alarm_was_confirmed* confirms the silent alarm

Based on the potential feature combination *Manual Power Window* and *LED Power Window*, the standard specification is extended such that the release of the buttons for the window movement controls the corresponding LEDs. The new interface is shown in Fig. 3.26, defining further input and output signals.

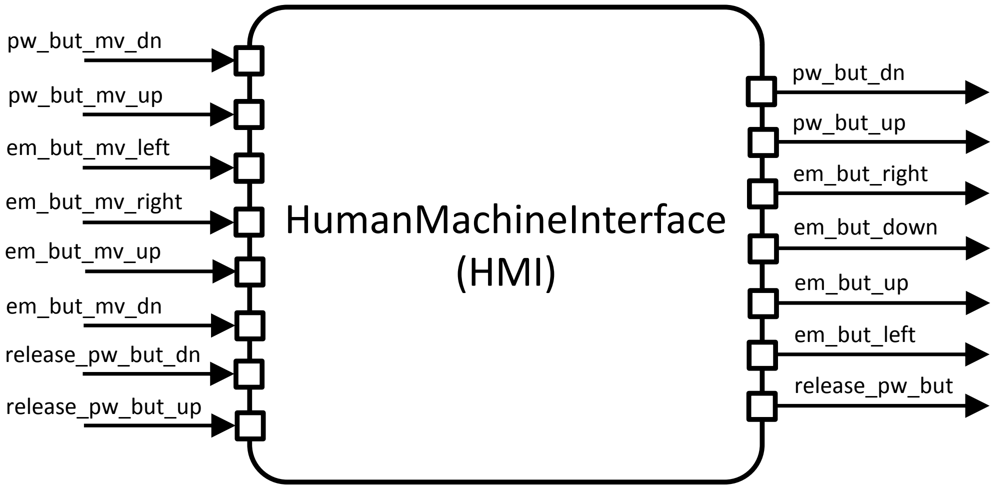

Figure 3.26.: Interface Specification of the HumanMachineInterface Component Variant with Manual Power Window and LED Power Window Features

Input Signals:

- *release_pw_but_dn* represents the releasing signal of the window down button
- *release_pw_but_up* represents the releasing signal of the window up button

Output Signals:

- *release_pw_but* represents the release of a window button

Based on the potential feature combination *Manual Power Window*, *LED Power Window* and *Alarm System*, another component variant is defined as depicted in Fig. 3.27, combining the signals of the standard specification and the signals of previously defined variants.

Based on the potential feature combination *Manual Power Window*, *LED Power Window*, *Alarm System* and *LED Alarm System*, another component variant is defined as shown in Fig. 3.28, combining the signals of the standard specification and the signals of previously defined variants.

LED Manual Power Window The first standard component for the feature *LED Power Window* is depicted in Fig. 3.29, controlling the turning on and off of one LED for the upwards movement and one LED for the downwards movement of the manual power window. There are no further component variants. The interface is defined by the following input and output signals.

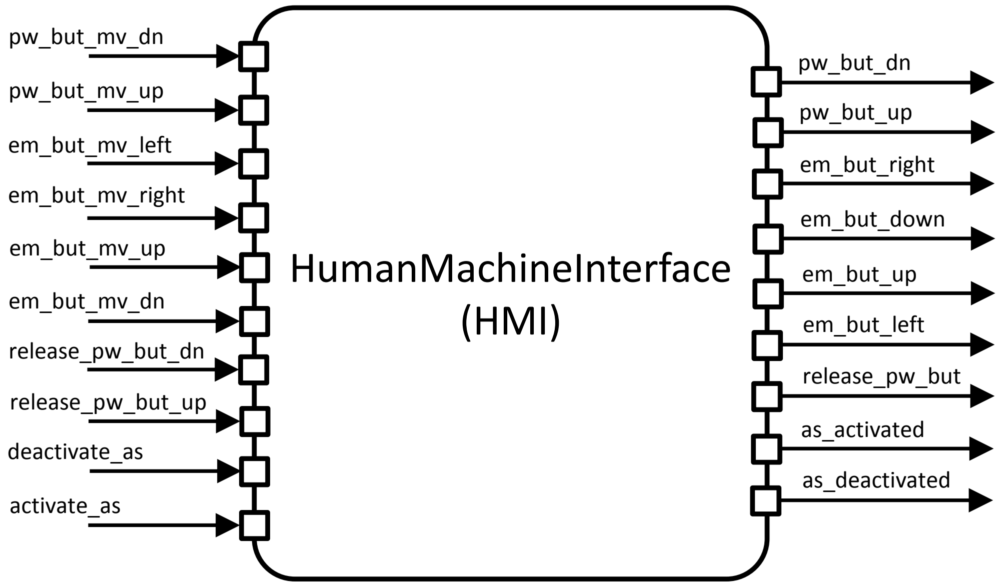

Figure 3.27.: Interface Specification of the HumanMachineInterface Component Variant with Manual Power Window, LED Power Window and Alarm System Features

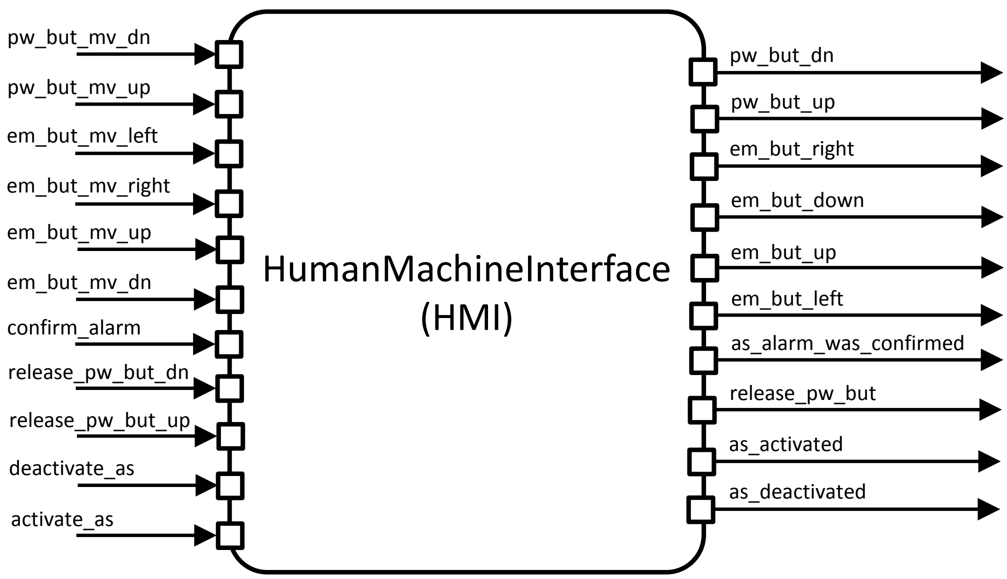

Figure 3.28.: Interface Specification of the HumanMachineInterface Component Variant with Manual Power Window, LED Power Window, Alarm System and LED Alarm System Features

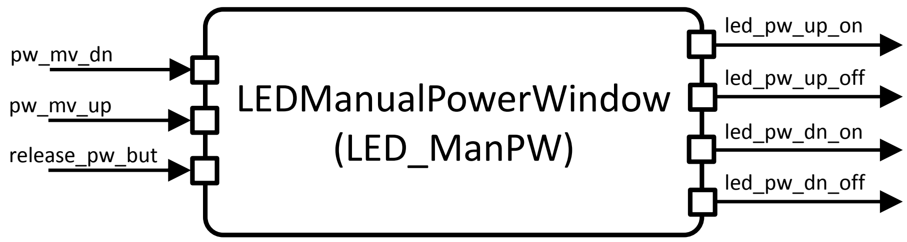

Figure 3.29.: Interface Specification of the LEDManualPowerWindow Component

Input Signals:

- *pw_mv_dn* controls the turning on of the LED for the upwards movement
- *pw_mv_up* controls the turning on of the LED for the downwards movement
- *release_pw_but* controls the turning off of the LEDs

Output Signals:

- *led_pw_up_on* switches on the LED for the upwards movement
- *led_pw_up_off* switches off the LED for the upwards movement
- *led_pw_dn_on* switches on the LED for the downwards movement
- *led_pw_dn_off* switches off the LED for the downwards movement

LED Automatic Power Window The second standard component for the feature *LED Power Window* is shown in Fig. 3.30, controlling the turning on and off of one LED for the upwards movement and one LED for the downwards movement of the automatic power window. The interface is defined by the following input and output signals.

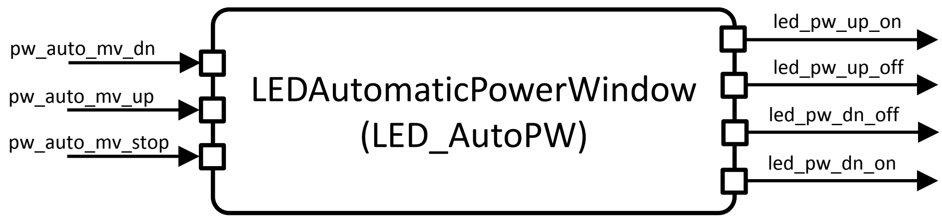

Figure 3.30.: Interface Specification of the Standard LEDAutomaticPowerWindow Component

Input Signals:

- *pw_auto_mv_dn* controls the turning on of the LED for the downwards movement
- *pw_auto_mv_up* controls the turning on of the LED for the upwards movement
- *pw_auto_mv_stop* controls the turning off of the LEDs

Output Signals:

- *led_pw_up_on* switches on the LED for the upwards movement
- *led_pw_up_off* switches off the LED for the upwards movement
- *led_pw_dn_on* switches on the LED for the downwards movement
- *led_pw_dn_off* switches off the LED for the downwards movement

Based on the feature *Central Locking System*, the standard specification is extended such that another LED exists being active when the window moves up and the car is locked. The new interface is depicted in Fig. 3.31, defining further input and output signals.

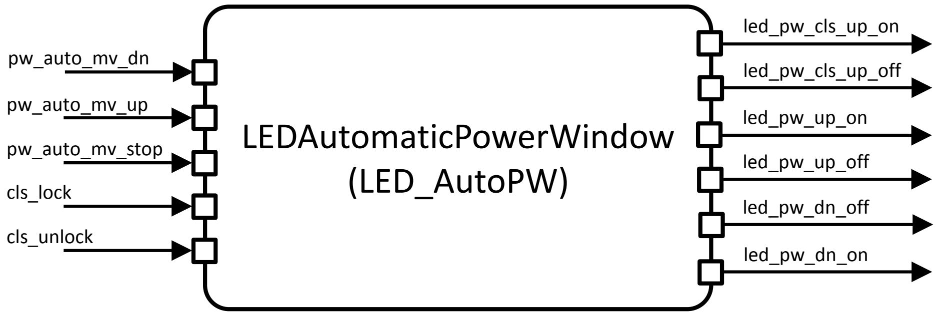

Figure 3.31.: Interface Specification of the LEDAutomaticPowerWindow Component Variant with Central Locking System Feature

Input Signals:

- *cls_lock* controls the turning on of the new LED for the upwards movement
- *cls_unlock* controls the turning off of the new LED for the upwards movement

Output Signals:

- *led_pw_cls_up_on* switches on the new LED for the upwards movement
- *led_pw_cls_up_off* switches off the new LED for the upwards movement

LED Exterior Mirror Top The first standard component for the feature *LED Exterior Mirror* is shown in Fig. 3.32, controlling the turning on and off of an LED when the exterior mirror reaches and stays in its upper position. There are no further component variants. The interface is defined by the following input and output signals.
Input Signals:

- *em_pos_vert_top* controls the turning on of the LED
- *em_pos_vert_pend* controls the turning off of the LED

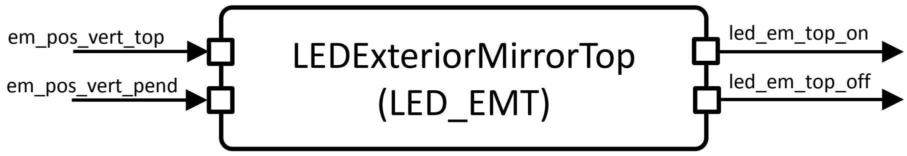

Figure 3.32.: Interface Specification of the LEDExteriorMirrorTop Component

Output Signals:

- *led_em_top_on* switches on the LED
- *led_em_top_off* switches off the LED

LED Exterior Mirror Left The second standard component for the feature *LED Exterior Mirror* is depicted in Fig. 3.33, controlling the turning on and off of an LED when the exterior mirror reaches and stays in its left-most position. There are no further component variants. The interface is defined by the following input and output signals.

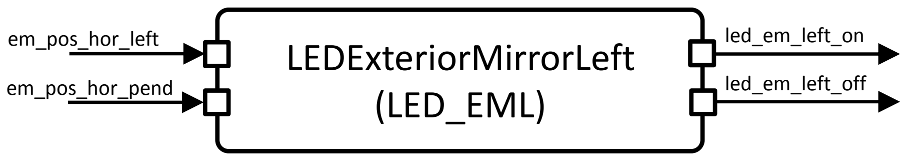

Figure 3.33.: Interface Specification of the LEDExteriorMirrorLeft Component

Input Signals:

- *em_pos_hor_left* controls the turning on of the LED
- *em_pos_hor_pend* controls the turning off of the LED

Output Signals:

- *led_em_left_on* switches on the LED
- *led_em_left_off* switches off the LED

LED Exterior Mirror Bottom The third standard component for the feature *LED Exterior Mirror* is shown in Fig. 3.34, controlling the turning on and off of an LED when the exterior mirror reaches and stays in its lower position. There are no further component variants. The interface is defined by the following input and output signals.
Input Signals:

- *em_pos_vert_bottom* controls the turning on of the LED
- *em_pos_vert_pend* controls the turning off of the LED

Output Signals:

- *led_em_bottom_on* switches on the LED
- *led_em_bottom_off* switches off the LED

Figure 3.34.: Interface Specification of the LEDExteriorMirrorBottom Component

LED Exterior Mirror Right The fourth standard component for the feature *LED Exterior Mirror* is depicted in Fig. 3.35, controlling the turning on and off of an LED when the exterior mirror reaches and stays in its right-most position. There are no further component variants. The interface is defined by the following input and output signals.

Figure 3.35.: Interface Specification of the LEDExteriorMirrorRight Component

Input Signals:

- *em_pos_hor_right* controls the turning on of the LED
- *em_pos_hor_pend* controls the turning off of the LED

Output Signals:

- *led_em_right_on* switches on the LED
- *led_em_right_off* switches off the LED

LED Exterior Mirror Heating The standard component for the feature *LED Heatable* is shown in Fig. 3.36, controlling the turning on and off of an LED representing the state of the mirror heater activation. There are no further component variants. The interface is defined by the following input and output signals.

Figure 3.36.: Interface Specification of the LEDExteriorMirrorHeating Component

Input Signals:

- *heating_on* controls the turning on of the LED
- *heating_off* controls the turning off of the LED

Output Signals:

- *led_em_heating_on* switches on the LED
- *led_em_heating_off* switches off the LED

LED Finger Protection The standard component for the feature *LED Finger Protection* is depicted in Fig. 3.37, controlling the turning on and off of an LED when the finger protection is active. There are no further component variants. The interface is defined by the following input and output signals.

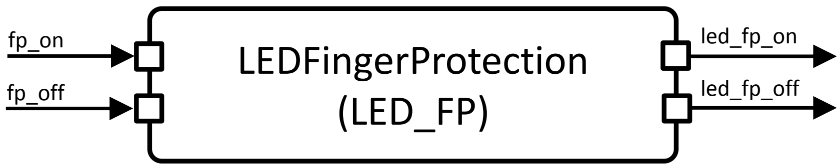

Figure 3.37.: Interface Specification of the LEDFingerProtection Component

Input Signals:

- *fp_on* controls the turning on of the LED
- *fp_off* controls the turning off of the LED

Output Signals:

- *led_fp_on* switches on the LED
- *led_fp_off* switches off the LED

LED Alarm System Active The first standard component for the feature *LED Alarm System* is shown in Fig. 3.38, controlling the turning on and off of an LED when the alarm system is enabled. There are no further component variants. The interface is defined by the following input and output signals.

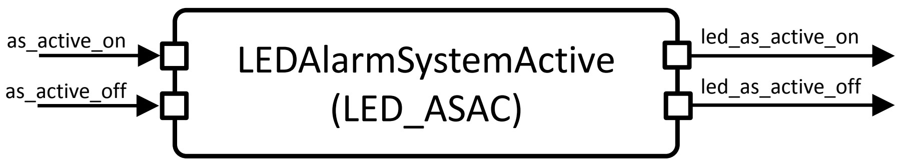

Figure 3.38.: Interface Specification of the LEDAlarmSystemActive Component

Input Signals:

- *as_active_on* controls the turning on of the LED
- *as_active_off* controls the turning off of the LED

Output Signals:

- *led_as_active_on* switches on the LED
- *led_as_active_off* switches off the LED

LED Alarm System Alarm The second standard component for the feature *LED Alarm System* is depicted in Fig. 3.39, controlling the turning on and off of an LED when the alarm of the alarm system is triggered. There are no further component variants. The interface is defined by the following input and output signals.

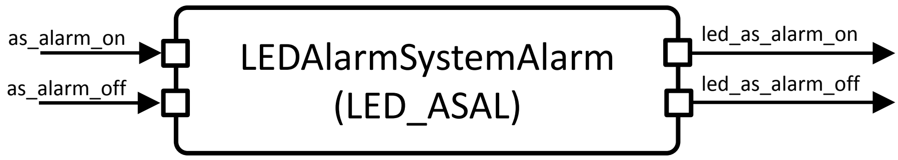

Figure 3.39.: Interface Specification of the LEDAlarmSystemAlarm Component

Input Signals:

- *as_alarm_on* controls the turning on of the LED
- *as_alarm_off* controls the turning off of the LED

Output Signals:

- *led_as_alarm_on* switches on the LED
- *led_as_alarm_off* switches off the LED

LED Alarm System Alarm Detected The third standard component for the feature *LED Alarm System* is shown in Fig. 3.40, controlling the turning on and off of an LED when the silent alarm is triggered, i.e., an alarm was detected and the alarm signal stops after the alarm time elapsed. There are no further component variants. The interface is defined by the following input and output signals.

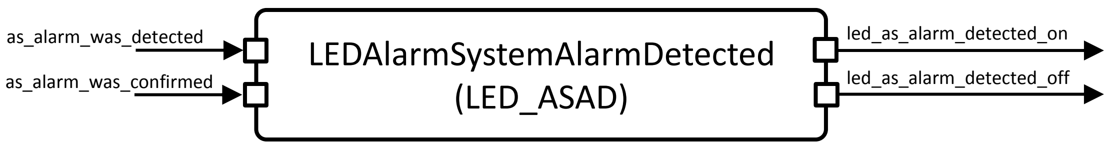

Figure 3.40.: Interface Specification of the LEDAlarmSystemAlarmDetected Component

Input Signals:

- *as_alarm_was_detected* controls the turning on of the LED
- *as_alarm_was_confirmed* controls the turning off of the LED after the driver confirmed the silent alarm

Output Signals:

- *led_as_alarm_detected_on* switches on the LED
- *led_as_alarm_detected_off* switches off the LED

LED Alarm System Interior Monitoring The standard component for the feature combination *LED Alarm System* and *Interior Monitoring* is depicted in Fig. 3.41, controlling the turning on and off of an LED when the interior alarm is triggered by the interior monitoring system. There are no further component variants. The interface is defined by the following input and output signals.

Figure 3.41.: Interface Specification of the LEDAlarmSystemInteriorMonitoring Component

Input Signals:

- *as_im_alarm_on* controls the turning on of the LED
- *as_im_alarm_off* controls the turning off of the LED

Output Signals:

- *led_as_im_alarm_on* switches on the LED
- *led_as_im_alarm_off* switches off the LED

LED Central Locking System The standard component for the feature *LED Central Locking System* is shown in Fig. 3.42, controlling the turning on and off of an LED representing the locking/unlocking state of the car. There are no further component variants. The interface is defined by the following input and output signals.

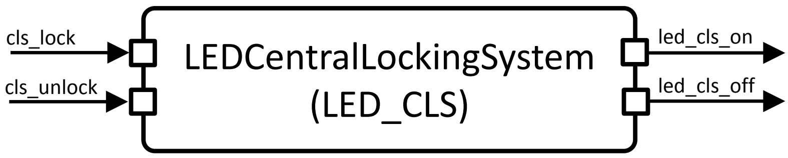

Figure 3.42.: Interface Specification of the LEDCentralLockingSystem Component

Input Signals:

- *cls_lock* controls the turning on of the LED
- *cls_unlock* controls the turning off of the LED

Output Signals:

- *led_cls_on* switches on the LED
- *led_cls_off* switches off the LED

Based on the variable component interfaces, we describe the deducible connector specifications in the next section.

3.2. Connector Specifications

The connector specifications exemplary shown in Fig. 3.1 are categorized in three types, namely *input*, *output* and *internal* connectors. Based on the interface specifications defined above, the internal connectors are directly deducible from matching input/output signals of the different component variants. For instance, the components *FingerProtection* and *ManualPowerWindow* communicate via a connector defined by the output signal *fp_on* and the input signal *fp_on*. Furthermore, we categorize the following input/output signals as input and output connectors as depicted in Tab. 3.1 and Tab. 3.2 for the communication with the system environment, where the listing is partitioned based on the corresponding features.

We use the defined component and connector specifications in the next section for the documentation of the architecture model deltas used for the definition of the architecture model variants documented in Sect. 3.4.

	Input Connectors	Output Connectors
HMI	pw_but_mv_dn pw_but_mv_up em_but_mv_left em_but_mv_right em_but_mv_up em_but_mv_dn	
HMI & AS	deactivate_as activate_as	
HMI & ManPW & LED PW	release_pw_but_up release_pw_but_dn	
HMI & LED AS	confirm_alarm	
Power Window	pw_pos_dn pw_pos_up	
Manual PW		pw_mv_dn pw_mv_up
Automatic PW		pw_auto_mv_up pw_auto_mv_dn pw_auto_mv_stop
Finger Protection	finger_detected	
Exterior Mirror	em_pos_left em_pos_right em_pos_top em_pos_bottom	em_mv_left em_mv_right em_mv_up em_mv_down
EM & Heatable	em_too_cold time_heating_elapsed	heating_on heating_off
LED EM		led_em_bottom_on led_em_bottom_off led_em_top_on led_em_top_off led_em_left_on led_em_left_off led_em_right_on led_em_right_off
LED Heatable		led_em_heating_on led_em_heating_off

Table 3.1.: Input and Output Connectors of the BCS Architecture Models (1)

	Input Connectors	Output Connectors
LED Power Window		led_pw_up_on led_pw_up_off led_pw_dn_on led_pw_dn_off
AutoPW & LED PW & CLS		led_pw_cls_up_on led_pw_cls_up_off
LED CLS		led_cls_on led_cls_off
LED Finger Protection		led_fp_on led_fp_off
LED Alarm System		led_as_alarm_detected_on led_as_alarm_detected_off led_as_alarm_on led_as_alarm_off led_as_active_on led_as_active_off
LED AS & IM		led_as_im_alarm_on led_as_im_alarm_off
Central Locking System	key_pos_lock key_pos_unlock	cls_lock cls_unlock
Automatic Locking	car_drives	car_lock car_unlock
Safety Function	door_open time_rck_sf_elapsed	
Remote Control Key	rck_but_lock rck_but_unlock	
RCK & CAP	pw_rm_up pw_rm_dn	
Alarm System	time_alarm_elapsed as_alarm_detected	as_alarm_on as_alarm_off as_active_on as_active_off as_alarm_was_detected
Interior Monitoring	im_alarm_detected	as_im_alarm_on as_im_alarm_off

Table 3.2.: Input and Output Connectors of the BCS Architecture Models (2)

3.3. Architecture Model Deltas

In this section, we document the architecture model deltas transforming the BCS core architecture model illustrated in Fig. 3.43 (see Sect. 3.4, pp. 69 – 70 for the description) to the various architecture model variants of the corresponding representative subset of product variants. We use the variability-aware architecture description language DeltaRX being an adaption of the delta modeling approach [9] for delta specifications.

DeltaRX provides several syntactical constructs for the textual description of hierarchical architectures as well as for architecture model deltas. An architecture model delta is identified by a unique name. In addition, we require the declaration of an application condition being a boolean statement over feature variables, specified by the keyword `when`. Based on the application condition, each delta is mapped to a product configuration. Due to dependencies between delta applications, we are able to indicate those dependencies by referencing to the deltas which have to be applied in advance. This optional part is specified by the keyword `after`. The transformations of the core, i.e., the addition and removal of architecture elements are indicated by the following keywords (cf. Tab. 3.3). Please note that the

	Add	Remove
Component	`addcomponent`	`removecomponent`
Connector	`addconnector`	`removeconnector`
Signal	`addsignal`	`removesignal`

Table 3.3.: Overview of DeltaRX Transformation Keywords

definition of ports is negligible due to the implicit definition by connector specifications.

As already mentioned, the various architecture model deltas are defined to transform a core architecture model. We used the core architecture model $P0$ corresponding to the core product $P0$ illustrated in Fig. 3.43 and its DeltaRX representation in the appendix A.2. In total, we define 25 architecture model deltas as follows.

Architecture Model Delta DAutomaticPW We define the delta *DAutomaticPW* applied to the core if the feature *Automatic Power Window* is selected for a product configuration. The delta shown in List. 3.1 in DeltaRX syntax transforms the core such that the component *Manual Power Window* (ManPW) as well as its connectors are removed and replaced by the component *Automatic Power Window* (AutoPW) with new connector definitions. Ports are implicitly specified, i.e., deducible from the connector definitions. The delta does not require any other deltas to be executed first.

```
DAutomaticPW when 'Automatic Power Window' {
  removeconnector{
    fp1
    fp2
    pw1
```

```
    pw2
    hmi5
    hmi6
    env13
    env14
  }

  removecomponent {
    ManPW
  }

  removesignal {
    pw_mv_dn
    pw_mv_up
  }

  addsignal {
    pw_auto_mv_up boolean
    pw_auto_mv_dn boolean
    pw_auto_mv_stop boolean
  }

  addcomponent {
    AutoPW{}
  }

  addconnector {
    fpautopw1(FP,fp_on,fp_on,AutoPW)
    fpautopw2(FP,fp_off,fp_off,AutoPW)
    hmiautopw1(HMI,pw_but_up,pw_but_up,AutoPW)
    hmiautopw2(HMI,pw_but_dn,pw_but_dn,AutoPW)
    autopwenv1(AutoPW,pw_auto_mv_up,pw_auto_mv_up,ENV)
    autopwenv2(AutoPW,pw_auto_mv_dn,pw_auto_mv_dn,ENV)
    autopwenv3(AutoPW,pw_auto_mv_stop,
      pw_auto_mv_stop,ENV)
    envautopw1(ENV,pw_pos_up,pw_pos_up,AutoPW)
    envautopw2(ENV,pw_pos_dn,pw_pos_dn,AutoPW)
  }
}
```

Listing 3.1: Architecture Model Delta DAutomaticPW

Architecture Model Delta DHeatable We define the delta *DHeatable* applied to the core if the feature *Heatable* is selected for a product configuration. The delta shown in List. 3.2 in DeltaRx syntax transforms the core such that the component *Exterior Mirror* (EM) as well as its connectors are removed and replaced by the component *Exterior Mirror with heating* (EMH) functionality. Furthermore, new connectors are defined to connect the new component with its communication partners (*Human Machine Interface*, *Environment*). Ports are implicitly specified, i.e., deducible from the connector definitions. The delta does not require any other deltas to be executed first.

```
DHeatable when 'Heatable' {
  removeconnector {
    em1
    em2
    em3
    em4
    hmi1
    hmi2
    hmi3
    hmi4
    env7
    env8
    env9
    env10
  }

  removecomponent {
    EM
  }

  addsignal {
    heating_on boolean
    heating_off boolean
    time_heating_elapsed boolean
    em_too_cold boolean
  }

  addcomponent {
    EMH {}
  }

  addconnector {
```

```
    emhenv1(EMH,em_mv_left,em_mv_left,ENV)
    emhenv2(EMH,em_mv_right,em_mv_right,ENV)
    emhenv3(EMH,em_mv_up,em_mv_up,ENV)
    emhenv4(EMH,em_mv_down,em_mv_down,ENV)
    emhenv5(EMH,heating_on,heating_on,ENV)
    emhenv6(EMH,heating_off,heating_off,ENV)
    envemh1(ENV,em_pos_right,em_pos_right,EMH)
    envemh2(ENV,em_pos_top,em_pos_top,EMH)
    envemh3(ENV,em_pos_bottom,em_pos_bottom,EMH)
    envemh4(ENV,em_too_cold,em_too_cold,EMH)
    envemh5(ENV,time_heating_elapsed,
      time_heating_elapsed,EMH)
    envemh6(ENV,em_pos_left,em_pos_left,EMH)

    hmiemh1(HMI,em_but_right,em_but_right,EMH)
    hmiemh2(HMI,em_but_left,em_but_left,EMH)
    hmiemh3(HMI,em_but_up,em_but_up,EMH)
    hmiemh4(HMI,em_but_down,em_but_down,EMH)
  }
}
```

Listing 3.2: Architecture Model Delta DHeatable

Architecture Model Delta DAS We define the delta *DAS* applied to the core if the feature *Alarm System* is selected for a product configuration. The delta shown in List. 3.3 in DeltaRx syntax transforms the core such that the component *Alarm System* (AS) is added to the architecture. Furthermore, new connectors are defined to connect the new component with its communication partners (*Human Machine Interface*, *Environment*). Ports are implicitly specified, i.e., deducible from the connector definitions. The delta does not require any other deltas to be executed first.

```
DAS when 'Alarm System' {
  addsignal {
    as_activated boolean
    as_deactivated boolean
    as_alarm_detected boolean
    time_alarm_elapsed boolean
    key_pos_lock boolean
    key_pos_unlock boolean
    as_active_on boolean
    as_active_off boolean
    as_alarm_on boolean
```

```
    as_alarm_off boolean
    activate_as boolean
    deactivate_as boolean
    as_alarm_was_detected boolean
  }

  addcomponent {
    AS {}
  }

  addconnector {
    envhm1(ENV,activate_as,activate_as,HMI)
    envhm2(ENV,deactivate_as,deactivate_as,HMI)
    hmias1(HMI,as_activated,as_activated,AS)
    hmias1(HMI,as_deactivated,as_deactivated,AS)

    envas1(ENV,as_alarm_detected,
      as_alarm_detected,AS)
    envas2(ENV,time_alarm_elapsed,
      time_alarm_elapsed,AS)
    envas3(ENV,key_pos_lock,key_pos_lock,AS)
    envas4(ENV,key_pos_unlock,key_pos_unlock,AS)

    asenv1(AS,as_active_on,as_active_on,ENV)
    asenv2(AS,as_alarm_on,as_alarm_on,ENV)
    asenv3(AS,as_active_off,as_active_off,ENV)
    asenv4(AS,as_alarm_off,as_alarm_off,ENV)
    asenv5(AS,as_alarm_was_detected,
      as_alarm_was_detected,ENV)
  }
}
```

Listing 3.3: Architecture Model Delta DAS

Architecture Model Delta DASIM We define the delta *DASIM* applied to the core if the features *Alarm System* and *Interior Monitoring* are selected for a product configuration. The delta requires the application of the delta *DAS*. The delta shown in List. 3.4 in DeltaRx syntax transforms the already modified core such that new connectors are defined for the component *Alarm System* (AS) to enable the interior monitoring. Ports are implicitly specified, i.e., deducible from the connector definitions.

```
DASIM after DAS when 'Interior Monitoring' {
```

```
  addsignal {
    im_alarm_detected boolean
    as_im_alarm_on boolean
    as_im_alarm_off boolean
  }

  addconnector {
    envasim1(ENV,im_alarm_detected,
      im_alarm_detected,AS)
    asimenv1(AS,as_im_alarm_on,as_im_alarm_on,ENV)
    asimenv2(AS,as_im_alarm_off,as_im_alarm_off,ENV)
  }
}
```

Listing 3.4: Architecture Model Delta DASIM

Architecture Model Delta DCLSM We define the delta *DCLSM* applied to the core if the features *Central Locking System* and *Manual Power Window* are selected for a product configuration. The delta shown in List. 3.5 in Deltarx syntax transforms the core such that the component *Central Locking System* (CLS) is added to the architecture. Furthermore, new connectors are defined to connect the new component with its communication partners (*Manual Power Window*, *Environment*). Ports are implicitly specified, i.e., deducible from the connector definitions. The delta does not require any other deltas to be executed first.

```
DCLSM when 'Central Locking System AND
  Manual Power Window' {
  addsignal {
    key_pos_lock boolean
    key_pos_unlock boolean
    cls_lock boolean
    cls_unlock boolean
  }

  addcomponent {
    CLS {}
  }

  addconnector {
    envcls1(ENV,key_pos_lock,key_pos_lock,CLS)
    envcls2(ENV,key_pos_unlock,key_pos_unlock,CLS)
    clsenv1(CLS,cls_lock,cls_lock,ENV)
    clsenv2(CLS,cls_unlock,cls_unlock,ENV)
```

```
    clsmanpw1(CLS,cls_lock,cls_lock,ManPW)
    clsmanpw2(CLS,cls_unlock,cls_unlock,ManPW)
  }
}
```

Listing 3.5: Architecture Model Delta DCLSM

Architecture Model Delta DCLSA We define the delta *DCLSA* applied to the core if the features *Central Locking System* and *Automatic Power Window* are selected for a product configuration. The delta requires the application of the delta *DAutomaticPW*. The delta shown in List. 3.6 in Deltarx syntax transforms the modified core such that the component *Central Locking System* (CLS) is added to the architecture. Furthermore, new connectors are defined to connect the new component with its communication partners (*Automatic Power Window*, *Environment*). Ports are implicitly specified, i.e., deducible from the connector definitions.

```
DCLSA when 'Central Locking System AND
  Automatic Power Window' {
  addsignal {
    key_pos_lock boolean
    key_pos_unlock boolean
    cls_unlock boolean
    cls_lock boolean
  }

  addcomponent {
    CLS {}
  }

  addconnector {
    envcls1(ENV,key_pos_lock,key_pos_lock,CLS)
    envcls2(ENV,key_pos_unlock,key_pos_unlock,CLS)
    clsenv1(CLS,cls_lock,cls_lock,ENV)
    clsenv2(CLS,cls_unlock,cls_unlock,ENV)
    clsautopw1(CLS,cls_lock,cls_lock,AutoPW)
    clsautopw2(CLS,cls_unlock,cls_unlock,AutoPW)
  }
}
```

Listing 3.6: Architecture Model Delta DCLSA

Architecture Model Delta DRCKA We define the delta *DRCKA* applied to the core if the features *Remote Control Key*, *Central Locking System* and *Automatic Power Window* are selected for a product configuration. The delta requires the application of the delta *DCLSA*. The delta shown in List. 3.7 in DeltaRx syntax transforms the modified core such that the component *Remote Control Key Controller* (RCK_Ctrl) is added to the architecture. Furthermore, new connectors are defined to connect the new component with its communication partners (*Central Locking System*, *Environment*). Ports are implicitly specified, i.e., deducible from the connector definitions. Based on two distinct ways to define the central locking system, a distinction for the remote control key controller is made as well for the feature *Automatic Power Window*.

```
DRCKA after DCLSA when 'Remote Control Key AND
  Automatic Power Window' {
  addsignal {
    rck_but_lock boolean
    rck_but_unlock boolean
    rck_lock boolean
    rck_unlock boolean
  }

  addcomponent {
    RCK_Ctrl {}
  }

  addconnector {
    envrck1(ENV,rck_but_lock,rck_but_lock,RCK_Ctrl)
    envrck2(ENV,rck_but_unlock,
      rck_but_unlock,RCK_Ctrl)
    rckcls1(RCK_Ctrl,rck_lock,rck_lock,CLS)
    rckcls2(RCK_Ctrl,rck_unlock,rck_unlock,CLS)
  }
}
```

Listing 3.7: Architecture Model Delta DRCKA

Architecture Model Delta DRCKM We define the delta *DRCKM* applied to the core if the features *Remote Control Key*, *Central Locking System* and *Manual Power Window* are selected for a product configuration. The delta requires the application of the delta *DCLSM*. The delta shown in List. 3.8 in DeltaRx syntax transforms the modified core such that the component *Remote Control Key Controller* (RCK_Ctrl) is added to the architecture. Furthermore, new connectors are defined to connect the new component with its communication partners (*Central Locking System*, *Environment*). Ports are implicitly specified, i.e., deducible from the connector definitions. Based on two distinct ways to define the central locking system, a distinction for the remote control key controller is made as well for the feature *Manual Power Window*.

```
DRCKM after DCLSM when 'Remote Control Key AND
  Manual Power Window' {
  addsignal {
    rck_but_lock boolean
    rck_but_unlock boolean
    rck_lock boolean
    rck_unlock boolean
  }

  addcomponent {
    RCK_Ctrl {}
  }

  addconnector {
    envrck1(ENV,rck_but_lock,rck_but_lock,RCK_Ctrl)
    envrck2(ENV,rck_but_unlock,
      rck_but_unlock,RCK_Ctrl)
    rckcls1(RCK_Ctrl,rck_lock,rck_lock,CLS)
    rckcls2(RCK_Ctrl,rck_unlock,rck_unlock,CLS)
  }
}
```

Listing 3.8: Architecture Model Delta DRCKM

Architecture Model Delta DRCKSFA We define the delta *DRCKSFA* applied to the core if the features *Remote Control Key*, *Safety Function*, *Central Locking System* and *Automatic Power Window* are selected for a product configuration. The delta requires the application of the delta *DRCKA*. The delta shown in List. 3.9 in DeltaRx syntax transforms the modified core such that new connectors are defined for the component *Remote Control Key Controller* (RCK_Ctrl) to enable the safety locking of the car. Ports are implicitly specified, i.e., deducible from the connector definitions. Based on two distinct ways to define the remote control key controller, a distinction for the connector definitions is made as well for the feature *Automatic Power Window*.

```
DRCKSFA after DRCKA when 'Safety Function AND
  Automatic Power Window' {
  addsignal {
    door_open boolean
    time_rck_sf_elapsed boolean
      }

```

```
  addconnector {
    envrcksf1(ENV,door_open,door_open,RCK_Ctrl)
    envrcksf2(ENV,time_rck_sf_elapsed,
      time_rck_sf_elapsed,RCK_Ctrl)
  }
}
```

Listing 3.9: Architecture Model Delta DRCKSFA

Architecture Model Delta DRCKSFM We define the delta *DRCKSFM* applied to the core if the features *Remote Control Key*, *Safety Function*, *Central Locking System* and *Manual Power Window* are selected for a product configuration. The delta requires the application of the delta *DRCKM*. The delta shown in List. 3.10 in DeltaRx syntax transforms the modified core such that new connectors are defined for the component *Remote Control Key Controller* (RCK_Ctrl) to enable the safety locking of the car. Ports are implicitly specified, i.e., deducible from the connector definitions. Based on two distinct ways to define the remote control key controller, a distinction for the connector definitions is made as well for the feature *Manual Power Window*.

```
DRCKSFM after DRCKM when 'Safety Function AND
  Manual Power Window' {
  addsignal {
    door_open boolean
    time_rck_sf_elapsed boolean
  }

  addconnector {
    envrcksf1(ENV,door_open,door_open,RCK_Ctrl)
    envrcksf2(ENV,time_rck_sf_elapsed,
      time_rck_sf_elapsed,RCK_Ctrl)
  }
}
```

Listing 3.10: Architecture Model Delta DRCKSFM

Architecture Model Delta DRCKCAP We define the delta *DRCKCAP* applied to the core if the features *Remote Control Key*, *Control Automatic Power Window* and *Automatic Power Window* are selected for a product configuration. The delta requires the application of the delta *DRCKA*. The delta shown in List. 3.11 in DeltaRx syntax transforms the modified core such that new connectors are defined for the component *Remote Control Key Controller* (RCK_Ctrl) to enable the remote control of the window movement. Ports are implicitly specified, i.e., deducible from the connector definitions.

```
DRCKCAP after DRCKA when 'Control Automatic Power Window
  AND Automatic Power Window' {
  addsignal {
    pw_rm_up boolean
    pw_rm_dn boolean
  }

  addconnector {
    envrckcap1(ENV,pw_rm_up,pw_rm_up,RCK_Ctrl)
    envrckcap2(ENV,pw_rm_dn,pw_rm_dn,RCK_Ctrl)
    rckcapautopw1(RCK_Ctrl,pw_but_up,
      pw_but_up, AutoPW)
    rckcapautopw2(RCK_Ctrl,pw_but_dn,
      pw_but_dn, AutoPW)
    rckcapfp1(RCK_Ctrl,pw_but_dn,pw_but_dn,FP)
  }
}
```

Listing 3.11: Architecture Model Delta DRCKCAP

Architecture Model Delta DCASM We define the delta *DCASM* applied to the core if the features *Remote Control Key*, *Control Alarm System* and *Manual Power Window* are selected for a product configuration. The delta requires the application of the deltas *DAS* and *DRCKM*. The delta shown in List. 3.12 in Deltarx syntax transforms the modified core such that new connectors are defined for the component *Remote Control Key Controller* (RCK_Ctrl) to enable the remote control of the alarm system. Ports are implicitly specified, i.e., deducible from the connector definitions. Based on two distinct ways to define the remote control key controller, a distinction for the connector definitions is made as well for the feature *Manual Power Window*.

```
DCASM after DAS DRCKM when 'Control Alarm System
  AND Manual Power Window' {
  addsignal{
    rck_lock boolean
    rck_unlock boolean
  }

  addconnector {
    rckcasas1(RCK_Ctrl,rck_lock,rck_lock,AS)
    rckcasas2(RCK_Ctrl,rck_unlock,rck_unlock,AS)
  }
}
```

Listing 3.12: Architecture Model Delta DCASM

Architecture Model Delta DCASA We define the delta *DCASA* applied to the core if the features *Remote Control Key*, *Control Alarm System* and *Automatic Power Window* are selected for a product configuration. The delta requires the application of the deltas *DAS* and *DRCKA*. The delta shown in List. 3.13 in DeltaRx syntax transforms the modified core such that new connectors are defined for the component *Remote Control Key Controller* (RCK_Ctrl) to enable the remote control of the alarm system. Ports are implicitly specified, i.e., deducible from the connector definitions. Based on two distinct ways to define the remote control key controller, a distinction for the connector definitions is made as well for the feature *Automatic Power Window*.

```
DCASA after DAS DRCKA when 'Control Alarm System
  AND Automatic Power Window' {
  addsignal{
    rck_lock boolean
    rck_unlock boolean
  }

  addconnector {
    rckcasas1(RCK_Ctrl,rck_lock,rck_lock,AS)
    rckcasas2(RCK_Ctrl,rck_unlock,rck_unlock,AS)
  }
}
```

Listing 3.13: Architecture Model Delta DCASA

Architecture Model Delta DALA We define the delta *DALA* applied to the core if the features *Central Locking System*, *Automatic Locking* and *Automatic Power Window* are selected for a product configuration. The delta requires the application of the delta *DCLSA*. The delta shown in List. 3.14 in DeltaRx syntax transforms the modified core such that new connectors are defined for the component *Central Locking System* (CLS) to enable the automatic locking of the car when the car is driving. Ports are implicitly specified, i.e., deducible from the connector definitions. Based on two distinct ways to define the central locking system, a distinction for the connector definitions is made as well for the feature *Automatic Power Window*.

```
DALA after DCLSA when 'Automatic Locking AND
  Automatic Power Window' {
  addsignal {
    door_open boolean
    car_drives boolean
```

```
    car_lock boolean
    car_unlock boolean
  }

  addconnector {
    envclsal1(ENV,car_drives,car_drives,CLS)
    envclsal2(ENV,door_open,door_open,CLS)
    clsalsenv1(CLS,car_lock,car_lock,ENV)
    clsalsenv2(CLS,car_unlock,car_unlock,ENV)
  }
}
```

Listing 3.14: Architecture Model Delta DALA

Architecture Model Delta DALM We define the delta *DALM* applied to the core if the features *Central Locking System*, *Automatic Locking* and *Manual Power Window* are selected for a product configuration. The delta requires the application of the delta *DCLSM*. The delta shown in List. 3.15 in DeltaRx syntax transforms the modified core such that new connectors are defined for the component *Central Locking System* (CLS) to enable the automatic locking of the car when the car is driving. Ports are implicitly specified, i.e., deducible from the connector definitions. Based on two distinct ways to define the central locking system, a distinction for the connector definitions is made as well for the feature *Manual Power Window*.

```
DALM after DCLSM when 'Automatic Locking AND
  Manual Power Window' {
  addsignal {
    door_open boolean
    car_drives boolean
    car_lock boolean
    car_unlock boolean
  }
  addconnector {
    envclsal1(ENV,car_drives,car_drives,CLS)
    envclsal2(ENV,door_open,door_open,CLS)
    clsalsenv1(CLS,car_lock,car_lock,ENV)
    clsalsenv2(CLS,car_unlock,car_unlock,ENV)
  }
}
```

Listing 3.15: Architecture Model Delta DALM

Architecture Model Delta DLEDEM We define the delta *DLEDEM* applied to the core if the feature *LED Exterior Mirror* is selected for a product configuration. The delta shown in List. 3.16 in DeltaRx syntax transforms the core such that the components *LED Exterior Mirror Bottom* (LED_EMB), *LED Exterior Mirror Top* (LED_EMT), *LED Exterior Mirror Right* (LED_EMR) and *LED Exterior Mirror Left* (LED_EML) are added to the architecture. Furthermore, new connectors are defined to connect the new components with their communication partners (*Exterior Mirror, Environment*). Ports are implicitly specified, i.e., deducible from the connector definitions. The delta does not require any other deltas to be executed first.

```
DLEDEM when 'LED Exterior Mirror AND NOT Heatable' {
  addsignal {
    em_pos_vert_bottom boolean
    em_pos_vert_pend boolean
    em_pos_vert_top boolean

    em_pos_hor_right boolean
    em_pos_hor_pend boolean
    em_pos_hor_left boolean

    led_em_bottom_on boolean
    led_em_top_on boolean
    led_em_right_on boolean
    led_em_left_on boolean

    led_em_bottom_off boolean
    led_em_top_off boolean
    led_em_right_off boolean
    led_em_left_off boolean
  }

  addcomponent {
    LED_EMB {}
    LED_EMT {}
    LED_EMR {}
    LED_EML {}
  }

  addconnector {
    emledemb1(EM,em_pos_vert_bottom,
      em_pos_vert_bottom,LED_EMB)
    emledemb2(EM,em_pos_vert_pend,
      em_pos_vert_pend,LED_EMB)
```

```
    ledembenv1(LED_EMB,led_em_bottom_on,
      led_em_bottom_on,ENV)
    ledembenv2(LED_EMB,led_em_bottom_off,
      led_em_bottom_off,ENV)

    emledemt1(EM,em_pos_vert_pend,
      em_pos_vert_pend,LED_EMT)
    emledemt2(EM,em_pos_vert_top,
      em_pos_vert_top,LED_EMT)
    ledemtenv1(LED_EMT,led_em_top_on,
      led_em_top_on,ENV)
    ledemtenv2(LED_EMT,led_em_top_off,
      led_em_top_off,ENV)

    emledemr1(EM,em_pos_hor_right,
      em_pos_hor_right,LED_EMR)
    emledemr2(EM,em_pos_hor_pend,
      em_pos_hor_pend,LED_EMR)
    ledemrenv1(LED_EMR,led_em_right_on,
      led_em_right_on,ENV)
    ledemrenv2(LED_EMR,led_em_right_off,
      led_em_right_off,ENV)

    emledeml1(EM,em_pos_hor_left,
      em_pos_hor_left,LED_EML)
    emledeml2(EM,em_pos_hor_pend,
      em_pos_hor_pend,LED_EML)
    ledemlenv1(LED_EML,led_em_left_on,
      led_em_left_on,ENV)
    ledemlenv2(LED_EML,led_em_left_off,
      led_em_left_off,ENV)
  }
}
```

Listing 3.16: Architecture Model Delta DLEDEM

Architecture Model Delta DLEDEMH The delta *DLEDEMH* is applied to the core, if the feature *LED Exterior Mirror* has been selected and the *Heatable* feature has been selected. The delta is executed after the delta *DHeatable* has been applied. It is shown in List. 3.17 in DeltaRX syntax. It transforms the modified core such that the component *Exterior Mirror* (*EM*) is removed and replaced with textitExterior Mirror Heating (*EMH*). Additionally, it adds four new components representing the LEDs for the exterior mirror. Furthermore, new connectors are defined to connect the new components with their communication partners (*Exterior Mirror Heating*, *Environment*). Ports are implicitly specified, i.e., deducible from the connector definitions.

```
DLEDEMH after DHeatable when 'LED Exterior Mirror
  AND Heatable' {
  addsignal {
    em_pos_vert_bottom boolean
    em_pos_vert_pend boolean
    em_pos_vert_top boolean

    em_pos_hor_right boolean
    em_pos_hor_pend boolean
    em_pos_hor_left boolean

    led_em_bottom_on boolean
    led_em_bottom_off boolean
    led_em_top_on boolean
    led_em_top_off boolean
    led_em_right_on boolean
    led_em_right_off boolean
    led_em_left_on boolean
    led_em_left_off boolean
  }

  addcomponent {
    LED_EMB {}
    LED_EMT {}
    LED_EMR {}
    LED_EML {}
  }

  addconnector {
    emhledemb1(EMH,em_pos_vert_bottom,
      em_pos_vert_bottom,LED_EMB)
    emhledemb2(EMH,em_pos_vert_pend,
      em_pos_vert_pend,LED_EMB)
    ledembenv1(LED_EMB,led_em_bottom_on,
      led_em_bottom_on,ENV)
    ledembenv2(LED_EMB,led_em_bottom_off,
```

```
      led_em_bottom_off,ENV)

    emhledemt1(EMH,em_pos_vert_pend,
      em_pos_vert_pend,LED_EMT)
    emhledemt2(EMH,em_pos_vert_top,
      em_pos_vert_top,LED_EMT)
    ledemtenv1(LED_EMT,led_em_top_on,
      led_em_top_on,ENV)
    ledemtenv2(LED_EMT,led_em_top_off,
      led_em_top_off,ENV)

    emhledemr1(EMH,em_pos_hor_right,
      em_pos_hor_right,LED_EMR)
    emhledemr2(EMH,em_pos_hor_pend,
      em_pos_hor_pend,LED_EMR)
    ledemrenv1(LED_EMR,led_em_right_on,
      led_em_right_on,ENV)
    ledemrenv2(LED_EMR,led_em_right_off,
      led_em_right_off,ENV)

    emhledeml1(EMH,em_pos_hor_left,
      em_pos_hor_left,LED_EML)
    emhledeml2(EMH,em_pos_hor_pend,
      em_pos_hor_pend,LED_EML)
    ledemlenv1(LED_EML,led_em_left_on,
      led_em_left_on,ENV)
    ledemlenv2(LED_EML,led_em_left_off,
      led_em_left_off,ENV)
  }
}
```

Listing 3.17: Architecture Model Delta DLEDEMH

Architecture Model Delta DLEDHeatable We define the delta *DLEDHeatable* applied to the core if the feature *LED Heatable* is selected for a product configuration. The delta requires the application of the delta *DHeatable*. The delta shown in List. 3.18 in DeltaRx syntax transforms the modified core such that the component *LED Exterior Mirror Heating* (LED_EMH) is added to the architecture. Furthermore, new connectors are defined to connect the new component with its communication partners (*Exterior Mirror*, *Environment*). Ports are implicitly specified, i.e., deducible from the connector definitions.

```
DLEDHeatable after DHeatable when 'LED Heatable' {
```

```
  addsignal {
    led_em_heating_on boolean
    led_em_heating_off boolean
  }

  addcomponent {
    LED_EMH {}
  }

  addconnector {
    emhled1(EMH,heating_on,heating_on,LED_EMH)
    emhled2(EMH,heating_off,heating_off,LED_EMH)
    ledemhenv1(LED_EMH,led_em_heating_on,
      led_em_heating_on,ENV)
    ledemhenv2(LED_EMH,led_em_heating_off,
      led_em_heating_off,ENV)
  }
}
```

Listing 3.18: Architecture Model Delta DLEDHeatable

Architecture Model Delta DLEDFingerProtection We define the delta *DLED FingerProtection* applied to the core if the feature *LED Finger Protection* is selected for a product configuration. The delta shown in List. 3.19 in DeltaRx syntax transforms the core such that the component *LED Finger Protection* (LED_FP) is added to the architecture. Furthermore, new connectors are defined to connect the new component with its communication partners (*Finger Protection, Environment*). Ports are implicitly specified, i.e., deducible from the connector definitions. The delta does not require any other deltas to be executed first.

```
DLEDFingerProtection when 'LED Finger Protection' {
  addsignal{
    led_fp_on boolean
    led_fp_off boolean
  }

  addcomponent {
    LED_FP {}
  }

  addconnector {
    fp3(FP,fp_on,fp_on,LED_FP)
    fp4(FP,fp_off,fp_off,LED_FP)
```

```
    ledfp1(LED_FP, led_fp_on,led_fp_on,ENV)
    ledfp2(LED_FP, led_fp_off,led_fp_off,ENV)
  }
}
```

Listing 3.19: Architecture Model Delta DLEDFingerProtection

Architecture Model Delta DLEDCLSM We define the delta *DLEDCLSM* applied to the core if the features *LED Central Locking System*, *Central Locking System* and *Manual Power Window* are selected for a product configuration. The delta requires the application of the delta *DCLSM*. The delta shown in List. 3.20 in DELTARX syntax transforms the modified core such that the component *LED Central Locking System* (LED_CLS) is added to the architecture. Furthermore, new connectors are defined to connect the new component with its communication partners (*Central Locking System*, *Environment*). Ports are implicitly specified, i.e., deducible from the connector definitions. Based on two distinct ways to define the central locking system, a distinction for the corresponding LED is made as well for the feature *Manual Power Window*.

```
DLEDCLSM after DCLSM when 'LED Central Locking System
  AND Manual Power Window' {
  addsignal{
    cls_lock boolean
    cls_unlock boolean
    led_cls_on boolean
    led_cls_off boolean
  }

  addcomponent {
    LED_CLS {}
  }

  addconnector {
    clsledcls1(CLS,cls_lock,cls_lock,LED_CLS)
    clsledcls2(CLS,cls_unlock,cls_unlock,LED_CLS)
    ledclsenv1(LED_CLS,led_cls_on,led_cls_on,ENV)
    ledclsenv2(LED_CLS,led_cls_off,led_cls_off,ENV)
  }
}
```

Listing 3.20: Architecture Model Delta DLEDCLSM

Architecture Model Delta DLEDCLSA We define the delta *DLEDCLSA* applied to the core if the features *LED Central Locking System*, *Central Locking System* and *Automatic Power Window* are selected for a product configuration. The delta requires the application of the delta *DCLSA*. The delta shown in List. 3.21 in Deltarx syntax transforms the modified core such that the component *LED Central Locking System* (LED_CLS) is added to the architecture. Furthermore, new connectors are defined to connect the new component with its communication partners (*Central Locking System*, *Environment*). Ports are implicitly specified, i.e., deducible from the connector definitions. Based on two distinct ways to define the central locking system, a distinction for the corresponding LED is made as well for the feature *Automatic Power Window*.

```
DLEDCLSA after DCLSA when 'LED Central Locking System
  AND Automatic Power Window' {
  addsignal{
    cls_lock boolean
    cls_unlock boolean
    led_cls_on boolean
    led_cls_off boolean
  }

  addcomponent {
    LED_CLS {}
  }

  addconnector {
    clsledcls1(CLS,cls_lock,cls_lock,LED_CLS)
    clsledcls2(CLS,cls_unlock,cls_unlock,LED_CLS)
    ledclsenv1(LED_CLS,led_cls_on,led_cls_on,ENV)
    ledclsenv2(LED_CLS,led_cls_off,led_cls_off,ENV)
  }
}
```

Listing 3.21: Architecture Model Delta DLEDCLSA

Architecture Model Delta DLEDAPW We define the delta *DLEDAPW* applied to the core if the features *LED Power Window* and *Automatic Power Window* are selected for a product configuration. The delta requires the application of the delta *DAutomaticPW*. The delta shown in List. 3.22 in Deltarx syntax transforms the modified core such that the component *LED Automatic Power Window* (LED_AutoPW) is added to the architecture. Furthermore, new connectors are defined to connect the new component with its communication partners (*Automatic Power Window*, *Environment*). Ports are implicitly specified, i.e., deducible from the connector definitions.

```
DLEDAPW after DAutomaticPW when 'LED Power Window AND
  Automatic Power Window AND
```

```
  NOT Central Locking System' {
  addsignal {
    led_pw_up_on boolean
    led_pw_up_off boolean
    led_pw_dn_on boolean
    led_pw_dn_off boolean
    pw_auto_mv_up boolean
    pw_auto_mv_dn boolean
    pw_auto_mv_stop boolean
  }

  addcomponent {
    LED_AutoPW {  }
  }

  addconnector {
    autopwledapw1(AutoPW,pw_auto_mv_dn,
      pw_auto_mv_dn,LED_AutoPW)
    autopwledapw2(AutoPW,pw_auto_mv_up,
      pw_auto_mv_up,LED_AutoPW)
    autopwledapw3(AutoPW,pw_auto_mv_stop,
      pw_auto_mv_stop,LED_AutoPW)

    ledapwenv1(LED_AutoPW,led_pw_up_on,
      led_pw_up_on,ENV)
    ledapwenv2(LED_AutoPW,led_pw_up_off,
      led_pw_up_off,ENV)
    ledapwenv3(LED_AutoPW,led_pw_dn_on,
      led_pw_dn_on,ENV)
    ledapwenv4(LED_AutoPW,led_pw_dn_off,
      led_pw_dn_off,ENV)
  }
}
```

Listing 3.22: Architecture Model Delta DLEDAPW

Architecture Model Delta DLEDAPWCLS We define the delta *DLEDAPWCLS* applied to the core if the features *LED Power Window*, *Automatic Power Window* and *Central Locking System* are selected for a product configuration. The delta requires the application of the deltas *DAutomaticPW* and *DCLSA*. The delta shown in List. 3.23 in DeltaRx syntax transforms the modified core such that the component *LED Automatic Power Window* (LED_AutoPW) is added to the architecture. Furthermore, new connectors are defined to connect the new component with its communication partners (*Automatic Power Window*, *Central Locking System*, *Environment*). Ports are implicitly specified, i.e., deducible from the connector definitions.

```
DLEDAPWCLS after DAutomaticPW when 'LED Power Window AND
  Automatic Power Window AND Central Locking System' {
  addsignal {
    led_pw_up_on boolean
    led_pw_up_off boolean
    led_pw_dn_on boolean
    led_pw_dn_off boolean
    led_pw_cls_up_on boolean
    led_pw_cls_up_off boolean
  }

  addcomponent {
    LED_AutoPW {}
  }

  addconnector {
    autopwledapw1(AutoPW,pw_auto_mv_dn,
      pw_auto_mv_dn,LED_AutoPW)
    autopwledapw2(AutoPW,pw_auto_mv_up,
      pw_auto_mv_up,LED_AutoPW)
    autopwledapw3(AutoPW,pw_auto_mv_stop,
      pw_auto_mv_stop,LED_AutoPW)

    clsledapw1(CLS,cls_lock,cls_lock,LED_AutoPW)
    clsledapw2(CLS,cls_unlock,cls_unlock,LED_AutoPW)

    ledapwenv1(LED_AutoPW,led_pw_up_on,
      led_pw_up_on,ENV)
    ledapwenv2(LED_AutoPW,led_pw_dn_on,
      led_pw_dn_on,ENV)
    ledapwenv3(LED_AutoPW,led_pw_cls_up_on,
      led_pw_cls_up_on,ENV)
    ledapwenv4(LED_AutoPW,led_pw_up_off,
      led_pw_up_off,ENV)
    ledapwenv5(LED_AutoPW,led_pw_cls_up_off,
      led_pw_cls_up_off,ENV)
    ledapwenv6(LED_AutoPW,led_pw_dn_off,
```

```
      led_pw_dn_off,ENV)
  }
}
```

Listing 3.23: Architecture Model Delta DLEDAPWCLS

Architecture Model Delta DLEDManPW We define the delta *DLEDManPW* applied to the core if the features *Manual Power Window* and *LED Power Window* are selected for a product configuration. The delta shown in List. 3.24 in Deltarx syntax transforms the core such that the component *LED Manual Power Window* (LED_ManPW) is added to the architecture. Furthermore, new connectors are defined to connect the new component with its communication partners (*Manual Power Window*, *Environment*). Ports are implicitly specified, i.e., deducible from the connector definitions. The delta does not require any other deltas to be executed first.

```
DLEDManPW when 'Manual Power Window AND
  LED Power Window' {
  addsignal {
    release_pw_but boolean
    led_pw_up_on boolean
    led_pw_up_off boolean
    led_pw_dn_on boolean
    led_pw_dn_off boolean
    release_pw_but_dn boolean
    release_pw_but_up boolean
  }
  addcomponent {
    LED_ManPW {}
  }
  addconnector {
    manpwledapw1(ManPW,pw_mv_dn,
      pw_mv_dn,LED_ManPW)
    manpwledapw2(ManPW,pw_mv_up,
      pw_mv_up,LED_ManPW)
    hmiledapw3(HMI,release_pw_but,
      release_pw_but,LED_ManPW)

    envhmi1(ENV,release_pw_but_up,
      release_pw_but_up,HMI)
    envhmi2(ENV,release_pw_but_dn,
      release_pw_but_dn,HMI)

```

```
    ledmanpwenv1(LED_ManPW,led_pw_up_on,
      led_pw_up_on,ENV)
    ledmanpwenv2(LED_ManPW,led_pw_dn_on,
      led_pw_dn_on,ENV)
    ledmanpwenv3(LED_ManPW,led_pw_up_off,
      led_pw_up_off,ENV)
    ledmanpwenv4(LED_ManPW,led_pw_dn_off,
      led_pw_dn_off,ENV)
  }
}
```

Listing 3.24: Architecture Model Delta DLEDManPW

Architecture Model Delta DLEDAS We define the delta *DLEDAS* applied to the core if the features *Alarm System* and *LED Alarm System* are selected for a product configuration. The delta requires the application of the delta *DAS*. The delta shown in List. 3.25 in Deltarx syntax transforms the modified core such that the components *LED Alarm System Active* (LED_ASAC), *LED Alarm System Alarm Detected* (LED_ASAD) and *LED Alarm System Alarm* (LED_ASAL) are added to the architecture. Furthermore, new connectors are defined to connect the new components with their communication partners (*Alarm System*, *Human Machine Interface*, *Environment*). Ports are implicitly specified, i.e., deducible from the connector definitions.

```
DLEDAS after DAS when 'LED Alarm System' {
  removeconnector {
    asenv5
  }

  addsignal {
    led_as_active_on boolean
    led_as_active_off boolean
    led_as_alarm_on boolean
    led_as_alarm_off boolean
    led_as_alarm_detected_on boolean
    led_as_alarm_detected_off boolean
    as_alarm_was_confirmed boolean
    confirm_alarm boolean
  }

  addcomponent {
    LED_ASAD {}

    LED_ASAC {}
```

```

    LED_ASAL {}
  }

  addconnector {
    //ASAD
    hmiledasad1(HMI,as_alarm_was_confirmed,
      as_alarm_was_confirmed,LED_ASAD)
    asledasad1(AS,as_alarm_was_detected,
      as_alarm_was_detected,LED_ASAD)
    ledasadenv1(LED_ASAD,led_as_alarm_detected_on,
      led_as_alarm_detected_on,ENV)
    ledasadenv2(LED_ASAD,led_as_alarm_detected_off,
      led_as_alarm_detected_off,ENV)
    //ASAL
    asledasal1(AS,as_alarm_on,as_alarm_on,LED_ASAL)
    asledasal2(AS,as_alarm_off,as_alarm_off,LED_ASAL)
    ledasalenv1(LED_ASAL,led_as_alarm_on,
      led_as_alarm_on,ENV)
    ledasalenv2(LED_ASAL,led_as_alarm_off,
      led_as_alarm_off,ENV)
    //ASAC
    asledasac1(AS,as_active_on,as_active_on,LED_ASAC)
    asledasac2(AS,as_active_off,as_active_off,LED_ASAC)
    ledasacenv1(LED_ASAC,led_as_active_on,
      led_as_active_on,ENV)
    ledasacenv2(LED_ASAC,led_as_active_off,
      led_as_active_off,ENV)
    //HMI
    envhmi3(ENV,confirm_alarm,confirm_alarm,HMI)
  }
}
```

Listing 3.25: Architecture Model Delta DLEDAS

Architecture Model Delta DLEDASIM We define the delta *DLEDASIM* applied to the core if the features *Alarm System*, *LED Alarm System* and *Interior Monitoring* are selected for a product configuration. The delta requires the application of the deltas *DAS* and *DASIM*. The delta shown in List. 3.26 in DeltaRx syntax transforms the modified core such that the component *LED Alarm System Interior Monitoring* (LED_ASIM) is added to the architecture. Furthermore, new connectors are defined to connect the new component with its communication partners (*Alarm System*, *Environment*). Ports are implicitly specified, i.e., deducible from the connector definitions.

```
DLEDASIM after DAS DASIM when 'LED Alarm System
  AND Interior Monitoring' {
  addsignal {
    led_as_im_alarm_on boolean
    led_as_im_alarm_off boolean
  }

  addcomponent {
    LED_ASIM {}
  }

  addconnector {
    asledasim1(AS,as_im_alarm_on,
      as_im_alarm_on,LED_ASIM)
    asledasim2(AS,as_im_alarm_off,
      as_im_alarm_off,LED_ASIM)
    ledasimenv1(LED_ASIM,led_as_im_alarm_on,
      led_as_im_alarm_on,ENV)
    ledasimenv2(LED_ASIM,led_as_im_alarm_off,
      led_as_im_alarm_off,ENV)
  }
}
```

Listing 3.26: Architecture Model Delta DLEDASIM

In the following Tab. 3.4 and Tab. 3.5, we use these definitions to specify the mapping between the described deltas and the various product variants of the representative subset. Here, a cross in a column indicates that the corresponding delta has to be applied to the core to obtain the corresponding product variant. As product $P0$ is chosen as the core product, no deltas are defined.

Based on the delta definitions and their mapping, we describe their application and the resulting architecture model variants in the following section.

	$P0$	$P1$	$P2$	$P3$	$P4$	$P5$	$P6$	$P7$	$P8$
DAutomaticPW		X	X		X			X	X
DHeatable		X		X		X	X		X
DAS		X	X	X	X	X	X		
DASIM		X			X	X	X		
DCLSM				X			X		
DCLSA		X			X				
DRCKA		X	X		X				
DRCKM				X					
DRCKSFA			X		X				
DRCKSFM									
DRCKCAP			X		X				
DCASM				X					
DCASA		X	X						
DALA			X		X				
DALM							X		
DLEDEM				X		X	X		X
DLEDHeatable				X		X	X		X
DLEDFingerProtection		X				X	X		X
DLEDCLSM				X			X		
DLEDCLSA									
DLEDAPW									X
DLEDAPWCLS									
DLEDManPW				X		X	X		
DLEDAS				X		X	X		
DLEDASIM						X	X		

Table 3.4.: Mapping of Architecture Model Deltas to Product Variants $P0$-$P8$

	$P9$	$P10$	$P11$	$P12$	$P13$	$P14$	$P15$	$P16$	$P17$
DAutomaticPW	X			X		X	X		
DHeatable	X			X	X	X	X	X	X
DAS	X		X			X	X	X	
DASIM	X		X			X	X	X	
DCLSM			X		X			X	
DCLSA	X			X		X	X		
DRCKA	X			X		X	X		
DRCKM			X		X			X	
DRCKSFA	X			X		X			
DRCKSFM			X		X			X	
DRCKCAP	X			X		X	X		
DCASM			X					X	
DCASA	X					X	X		
DALA				X		X	X		
DALM			X					X	
DLEDEM	X	X						X	
DLEDHeatable	X				X			X	
DLEDFingerProtection	X	X	X		X			X	
DLEDCLSM			X		X			X	
DLEDCLSA	X								
DLEDAPW									
DLEDAPWCLS	X								
DLEDManPW			X						
DLEDAS	X		X					X	
DLEDASIM	X		X					X	

Table 3.5.: Mapping of Architecture Model Deltas to Product Variants $P9$-$P17$

3.4. Architecture Model Variants

In this section, we integrate the previously defined component variants, connector specifications and architecture model deltas for the documentation of the different architecture model variants of the representative subset of product variants ($P0$-$P17$).

Core Architecture Model P0 The architecture model $P0$ depicted in Fig. 3.43 is used as the core architecture model for the core product $P0$. Based on its feature configuration (cf. Tab. 2.1), the model consists of the following four components.

- Standard HumanMachineInterface (HMI) component
- Standard ManualPowerWindow (ManPW) component
- Standard FingerProtection (FP) component
- Standard Exterior Mirror (EM) component

Furthermore, the architecture model comprises 28 connectors divided into 13 input connectors and six output connectors (cf. Tab. 3.1, 3.2) as well as nine internal connectors deducible from the component interfaces. Due to the definition as core architecture model, there are no deltas for the model generation for $P0$.

Architecture Model P1 The architecture model variant $P1$ is shown in Fig. 3.44 and corresponds to product $P1$. Based on its feature configuration (cf. Tab. 2.1), the model consists of the following eight components.

- HumanMachineInterface (HMI) component variant with Alarm System feature
- ExteriorMirror (EMH) component variant with Heatable feature
- AutomaticPowerWindow (AutoPW) component variant with Remote Control Key feature
- Standard FingerProtection (FP) component
- Standard LEDFingerProtection (LED_FP) component
- Standard RemoteControlKeyController (RCK_Ctrl) component
- CentralLockingSystem (CLS) component variant with Central Locking System feature
- AlarmSystem (AS) component variant with Interior Monitoring feature

Furthermore, the architecture model comprises 65 connectors divided into 26 input connectors and 20 output connectors (cf. Tab. 3.1, 3.2) as well as 19 internal connectors deducible from the component interfaces. For the generation of the architecture model, we apply eight architecture model deltas to the core as shown in Tab. 3.4.

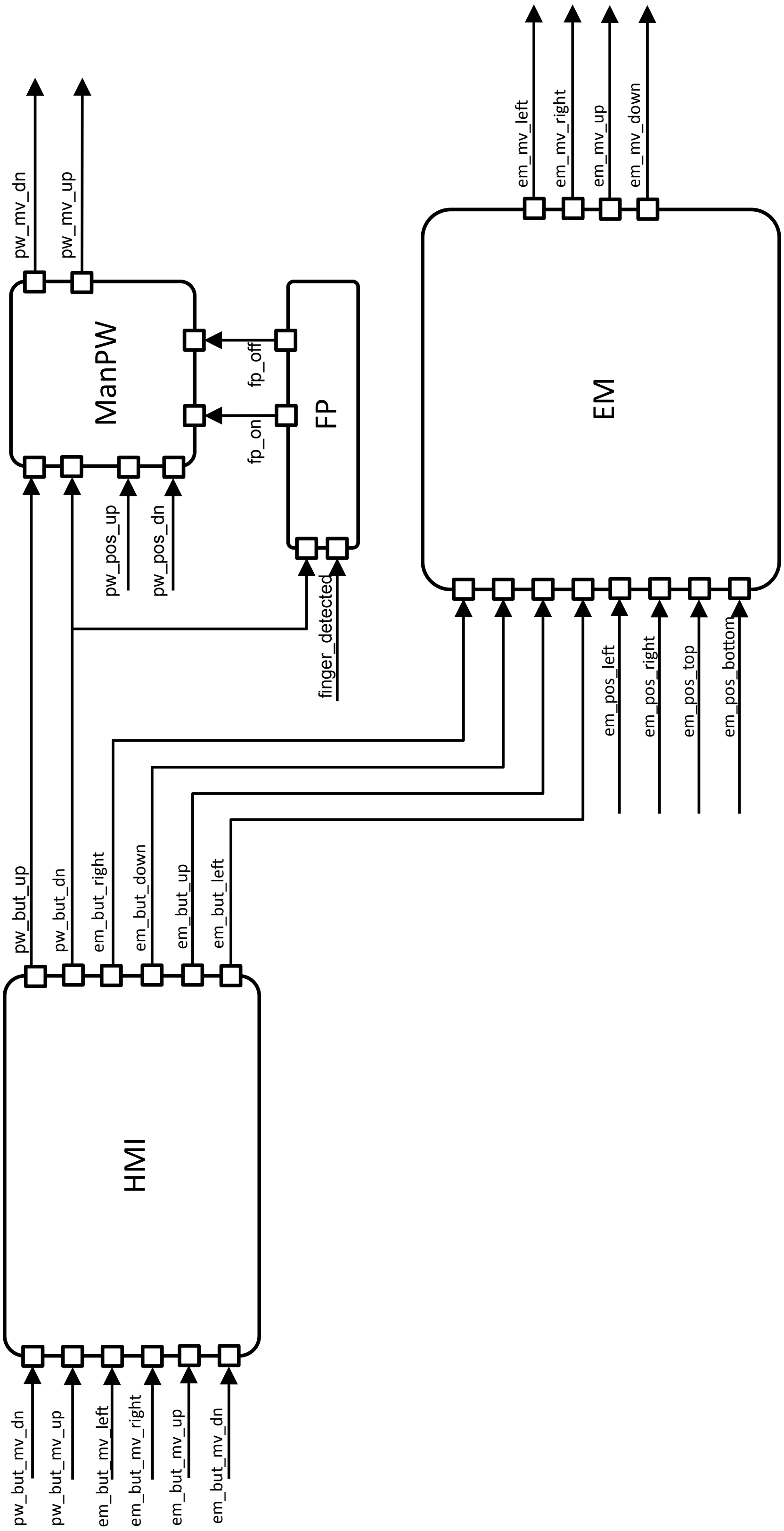

Figure 3.43.: Core Architecture Model for Core Product Po

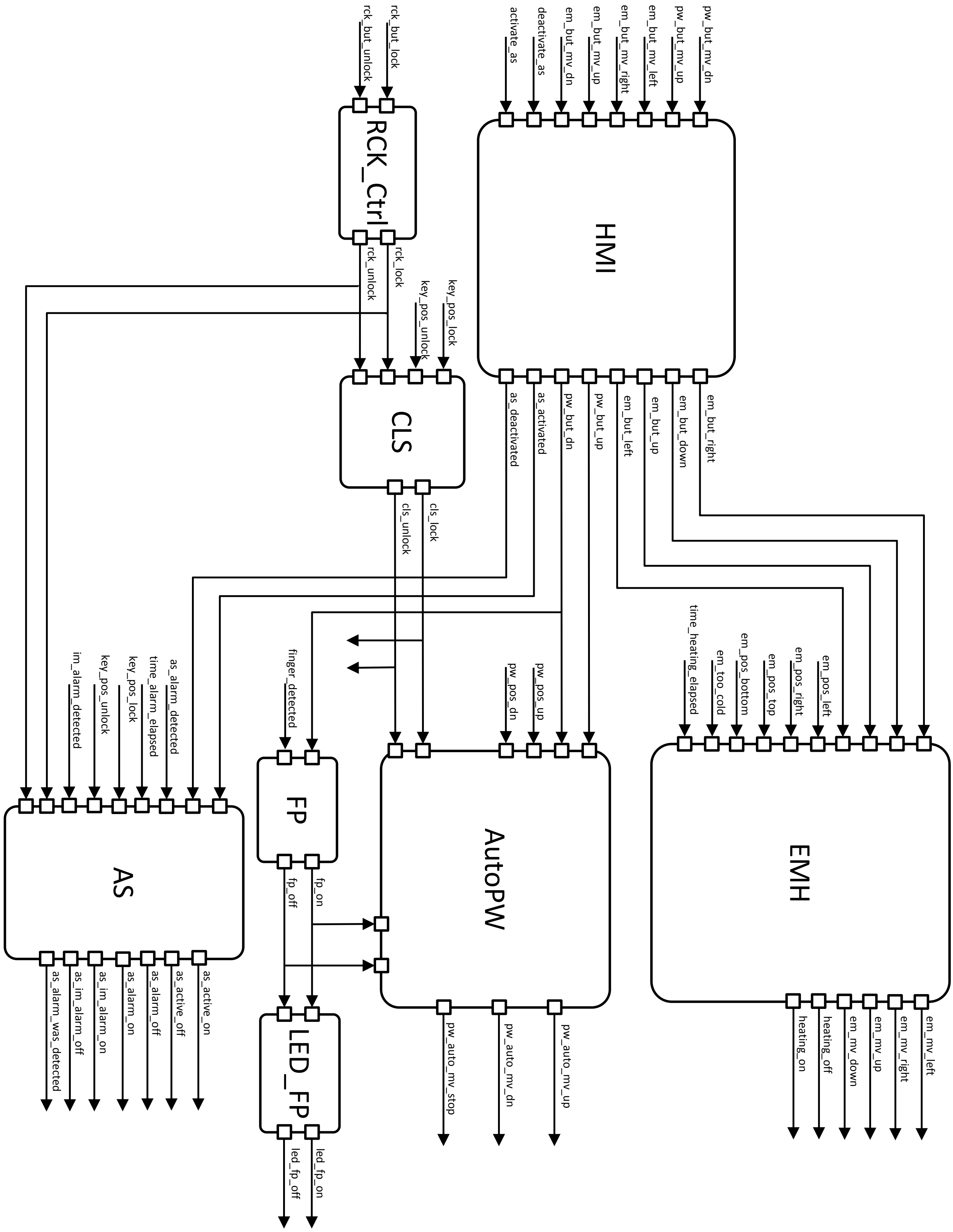

Figure 3.44.: Architecture Model Variant for Product P1

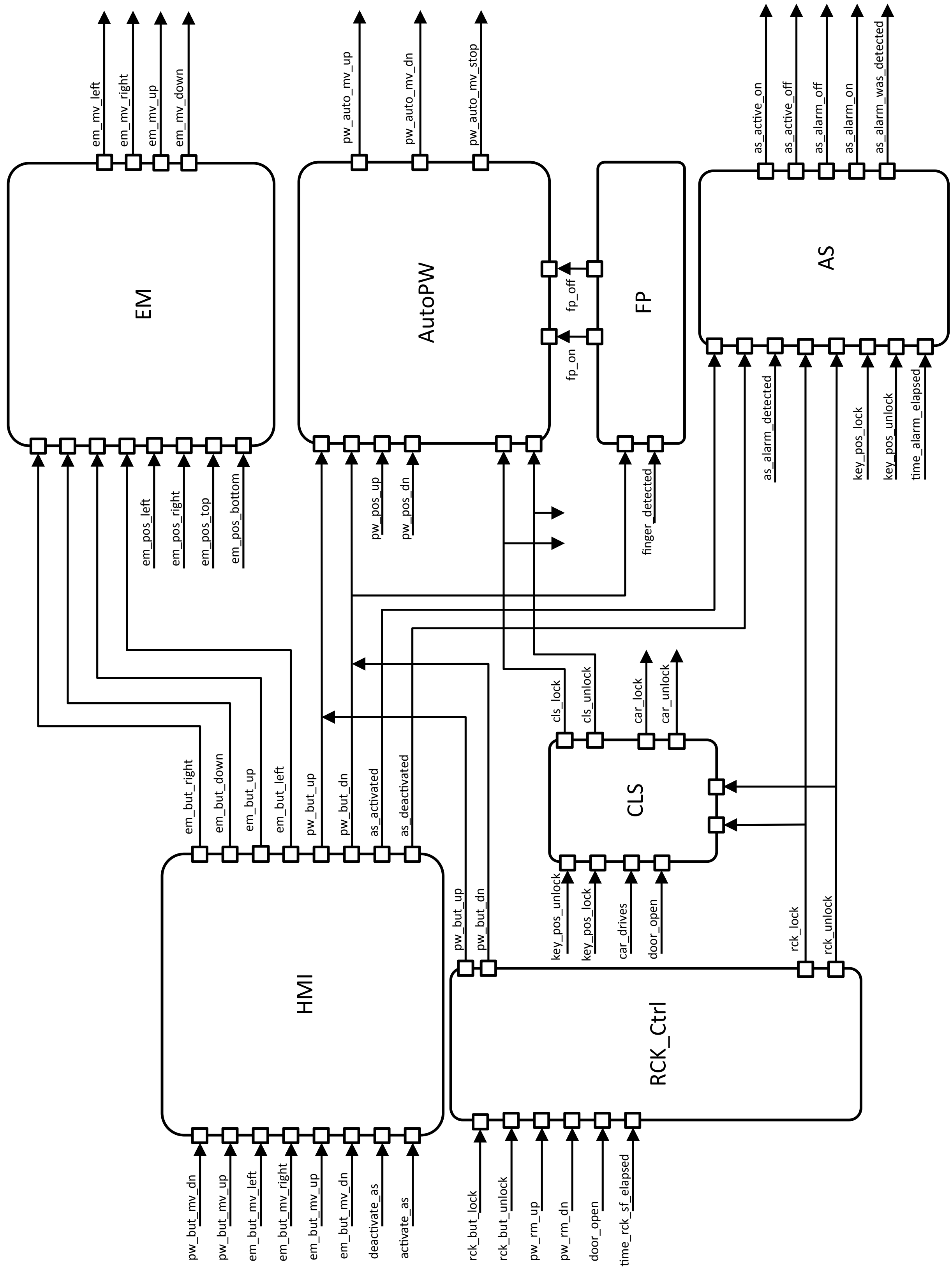

Figure 3.45.: Architecture Model Variant for Product P2

Architecture Model P2 The architecture model variant $P2$ is depicted in Fig. 3.45 and corresponds to product $P2$. Based on its feature configuration (cf. Tab. 2.1), the model consists of the following seven components.

- HumanMachineInterface (HMI) component variant with Alarm System feature
- Standard ExteriorMirror (EM) component
- AutomaticPowerWindow (AutoPW) component variant with Central Locking System feature
- Standard FingerProtection (FP) component
- RemoteControlKeyController (RCK_Ctrl) component variant with Control Automatic Power Window and Safety Function features
- CentralLockingSystem (CLS) component variant with Automatic Locking and Remote Control Key features
- AlarmSystem (AS) component variant with Control Alarm System feature

Furthermore, the architecture model comprises 64 connectors divided into 29 input connectors and 16 output connectors (cf. Tab. 3.1, 3.2) as well as 19 internal connectors deducible from the component interfaces. For the generation of the architecture model, we apply seven architecture model deltas to the core as shown in Tab. 3.4.

Architecture Model P3 The architecture model variant $P3$ is shown in Fig. 3.46 and corresponds to product $P3$. Based on its feature configuration (cf. Tab. 2.1), the model consists of the following 17 components.

- HumanMachineInterface (HMI) component variant with Alarm System, LED Power Window and LED Alarm System features
- ExteriorMirror (EMH) component variant with Heatable and LED Exterior Mirror features
- Standard LEDExteriorMirrorTop (LED_EMT) component
- Standard LEDExteriorMirrorLeft (LED_EML) component
- Standard LEDExteriorMirrorBottom (LED_EMB) component
- Standard LEDExteriorMirrorRight (LED_EMR) component
- Standard LEDExteriorMirrorHeating (LED_EMH) component
- ManualPowerWindow (ManPW) component variant with Central Locking System feature
- Standard LEDManualPowerWindow (LED_ManPW) component

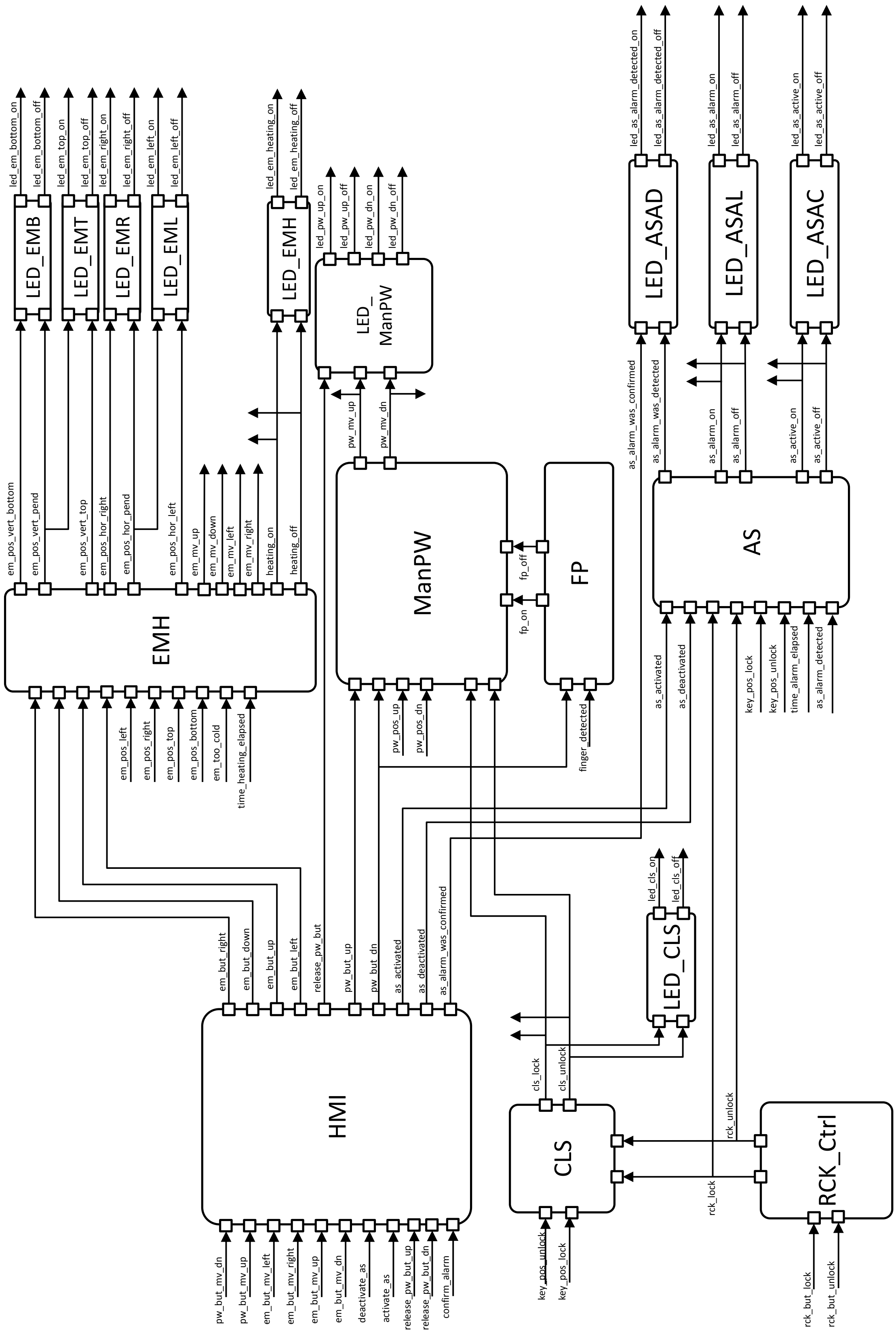

Figure 3.46.: Architecture Model Variant for Product P_3

- Standard FingerProtection (FP) component
- Standard RemoteControlKeyController (RCK_Ctrl) component
- CentralLockingSystem (CLS) component with Remote Control Key feature
- Standard LEDCentralLockingSystem (LED_CLS) component
- AlarmSystem (AS) component variant with Control Alarm System feature
- Standard LEDAlarmSystemAlarmDetected (LED_ASAD) component
- Standard LEDAlarmSystemAlarm (LED_ASAL) component
- Standard LEDAlarmSystemActive (LED_ASAC) component

Furthermore, the architecture model comprises 102 connectors divided into 28 input connectors and 36 output connectors (cf. Tab. 3.1, 3.2) as well as 38 internal connectors deducible from the component interfaces. For the generation of the architecture model, we apply 10 architecture model deltas to the core as shown in Tab. 3.4.

Architecture Model P4 The architecture model variant *P*4 is depicted in Fig. 3.47 and corresponds to product *P*4. Based on its feature configuration (cf. Tab. 2.1), the model consists of the following seven components.

- HumanMachineInterface (HMI) component variant with Alarm System feature
- Standard ExteriorMirror (EM) component
- AutomaticPowerWindow (AutoPW) component variant with Central Locking System feature
- Standard FingerProtection (FP) component
- RemoteControlKeyController (RCK_Ctrl) component variant with Control Automatic Power Window and Safety Function features
- CentralLockingSystem (CLS) component variant with Remote Control Key feature
- AlarmSystem (AS) component variant with Interior Monitoring feature

Furthermore, the architecture model comprises 64 connectors divided into 30 input connectors and 17 output connectors (cf. Tab. 3.1, 3.2) as well as 17 internal connectors deducible from the component interfaces. For the generation of the architecture model, we apply eight architecture model deltas to the core as shown in Tab. 3.4.

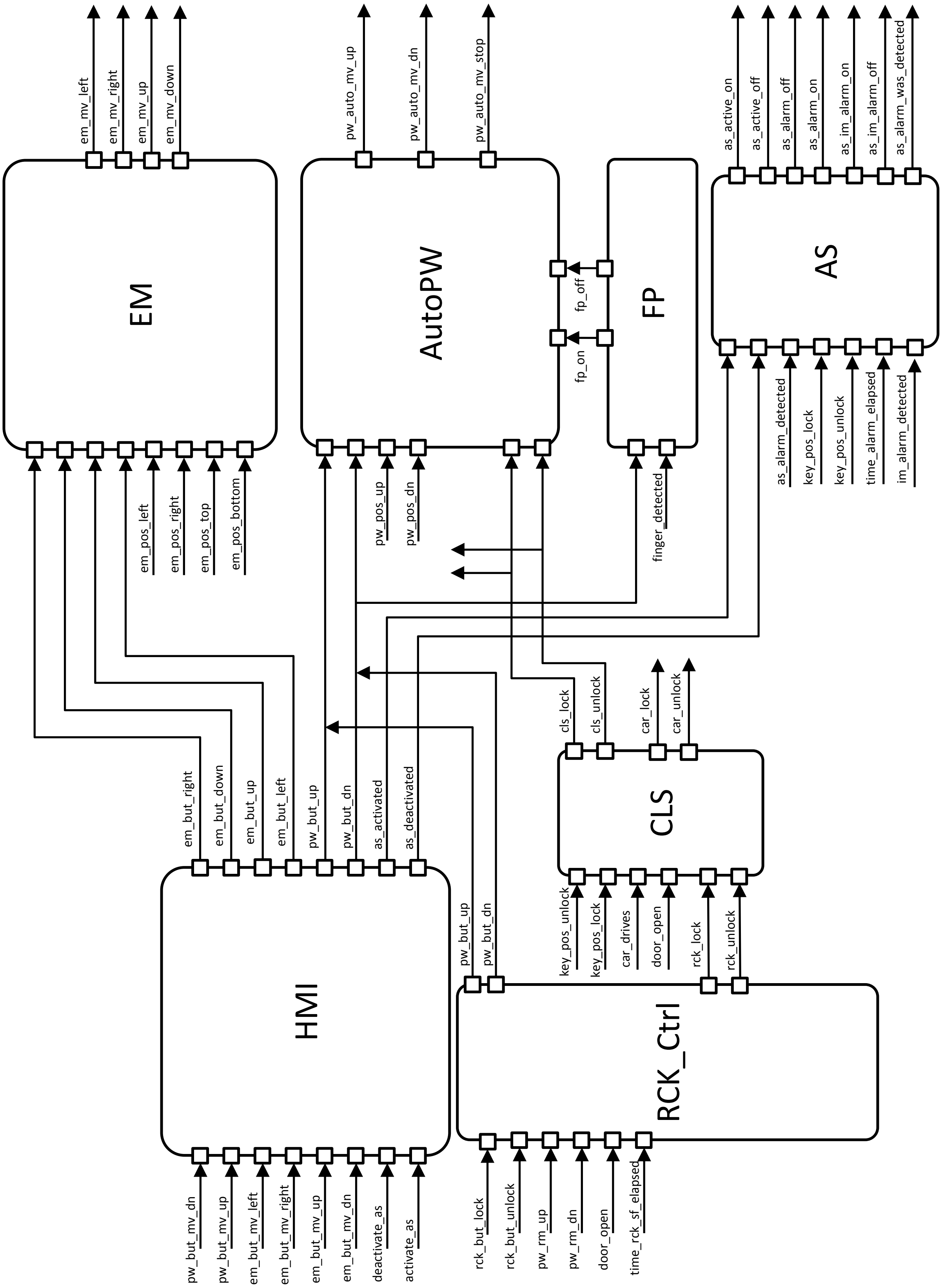

Figure 3.47.: Architecture Model Variant for Product P4

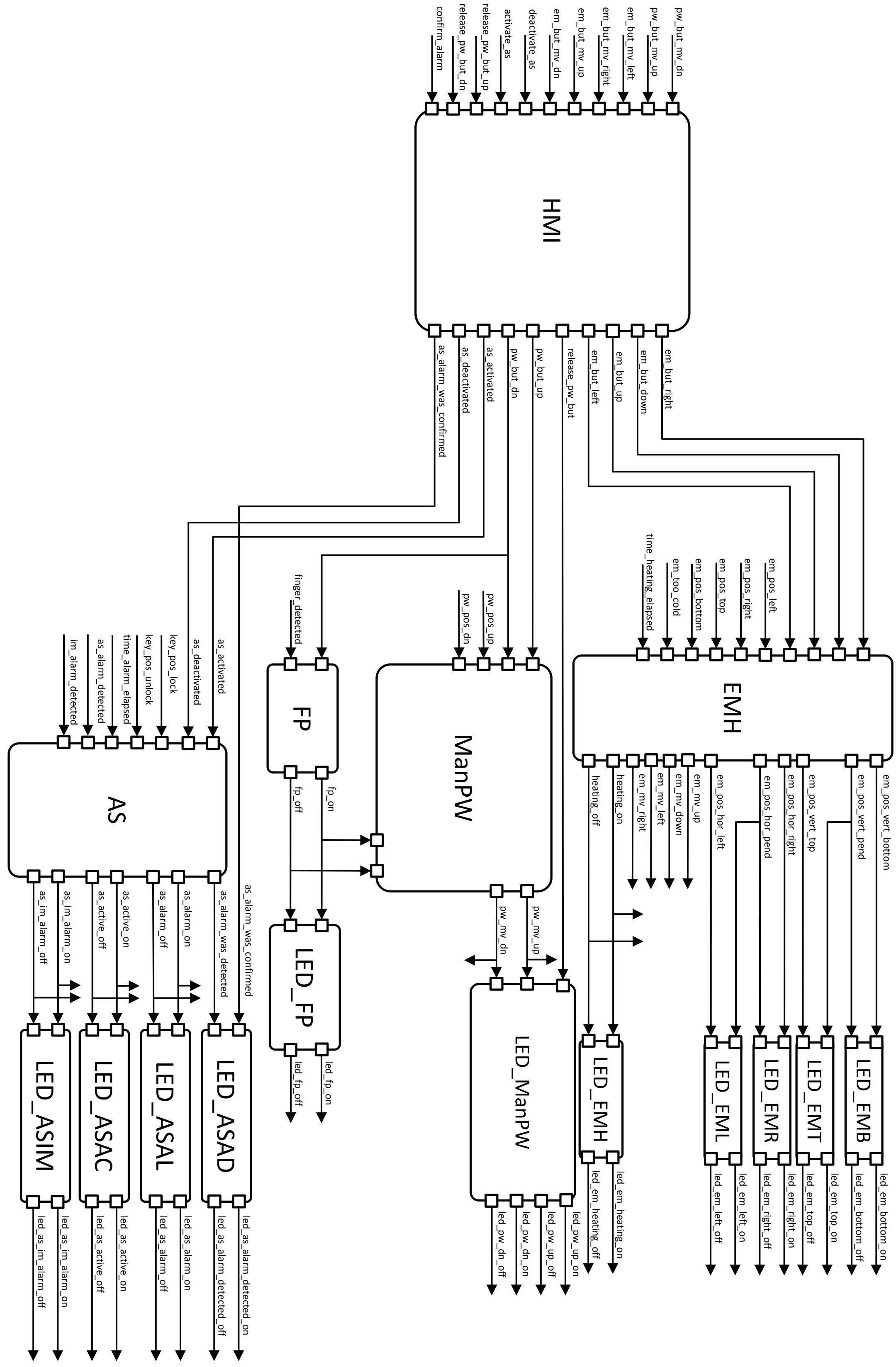

Figure 3.48.: Architecture Model Variant for Product P5

Architecture Model P5 The architecture model variant $P5$ is shown in Fig. 3.48 and corresponds to product $P5$. Based on its feature configuration (cf. Tab. 2.1), the model consists of the following 16 components.

- HumanMachineInterface (HMI) component variant with Alarm System and LED Power Window features
- ExteriorMirror (EMH) component variant with Heatable and LED Exterior Mirror features
- Standard LEDExteriorMirrorTop (LED_EMT) component
- Standard LEDExteriorMirrorLeft (LED_EML) component
- Standard LEDExteriorMirrorBottom (LED_EMB) component
- Standard LEDExteriorMirrorRight (LED_EMR) component
- Standard LEDExteriorMirrorHeating (LED_EMH) component
- Standard ManualPowerWindow (ManPW) component
- Standard LEDManualPowerWindow (LED_ManPW) component
- Standard FingerProtection (FP) component
- Standard LEDFingerProtection (LED_FP) component
- AlarmSystem (AS) component variant with Interior Monitoring feature
- Standard LEDAlarmSystemAlarmDetected (LED_ASAD) component
- Standard LEDAlarmSystemAlarm (LED_ASAL) component
- Standard LEDAlarmSystemActive (LED_ASAC) component
- Standard LEDAlarmSystemInteriorMonitoring (LED_ASIM) component

Furthermore, the architecture model comprises 96 connectors divided into 24 input connectors and 38 output connectors (cf. Tab. 3.1, 3.2) as well as 34 internal connectors deducible from the component interfaces. For the generation of the architecture model, we apply nine architecture model deltas to the core as shown in Tab. 3.4.

Architecture Model P6 The architecture model variant $P6$ is depicted in Fig. 3.49 and corresponds to product $P6$. Based on its feature configuration (cf. Tab. 2.1), the model consists of the following 18 components.

- HumanMachineInterface (HMI) component variant with Alarm System and LED Power Window features

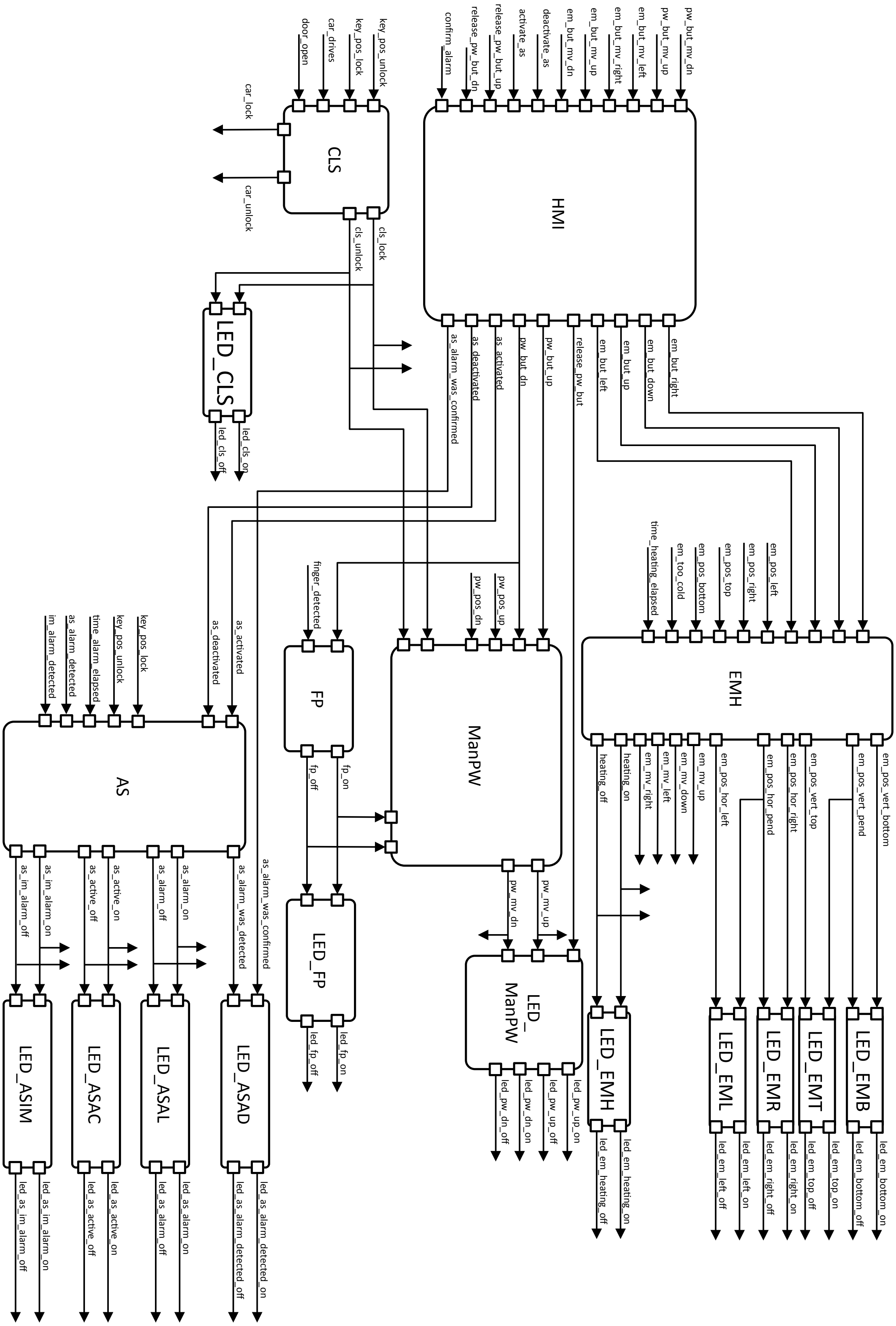

Figure 3.49.: Architecture Model Variant for Product P6

- ExteriorMirror (EMH) component variant with Heatable and LED Exterior Mirror features
- Standard LEDExteriorMirrorTop (LED_EMT) component
- Standard LEDExteriorMirrorLeft (LED_EML) component
- Standard LEDExteriorMirrorBottom (LED_EMB) component
- Standard LEDExteriorMirrorRight (LED_EMR) component
- Standard LEDExteriorMirrorHeating (LED_EMH) component
- ManualPowerWindow (ManPW) component variant with Central Locking System feature
- Standard LEDManualPowerWindow (LED_ManPW) component
- Standard FingerProtection (FP) component
- Standard LEDFingerProtection (LED_FP) component
- CentralLockingSystem (CLS) component variant with Automatic Locking feature
- Standard LEDCentralLockingSystem (LED_CLS) component
- AlarmSystem (AS) component variant with Interior Monitoring feature
- Standard LEDAlarmSystemAlarmDetected (LED_ASAD) component
- Standard LEDAlarmSystemAlarm (LED_ASAL) component
- Standard LEDAlarmSystemActive (LED_ASAC) component
- Standard LEDAlarmSystemInteriorMonitoring (LED_ASIM) component

Furthermore, the architecture model comprises 111 connectors divided into 29 input connectors and 44 output connectors (cf. Tab. 3.1, 3.2) as well as 38 internal connectors deducible from the component interfaces. For the generation of the architecture model, we apply 12 architecture model deltas to the core as shown in Tab. 3.4.

Architecture Model P7 The architecture model variant *P7* is shown in Fig. 3.50 and corresponds to product *P7*. Based on its feature configuration (cf. Tab. 2.1), the model consists of the following four components.

- Standard HumanMachineInterface (HMI) component
- Standard AutomaticPowerWindow (AutoPW) component
- Standard FingerProtection (FP) component
- Standard ExteriorMirror (EM) component

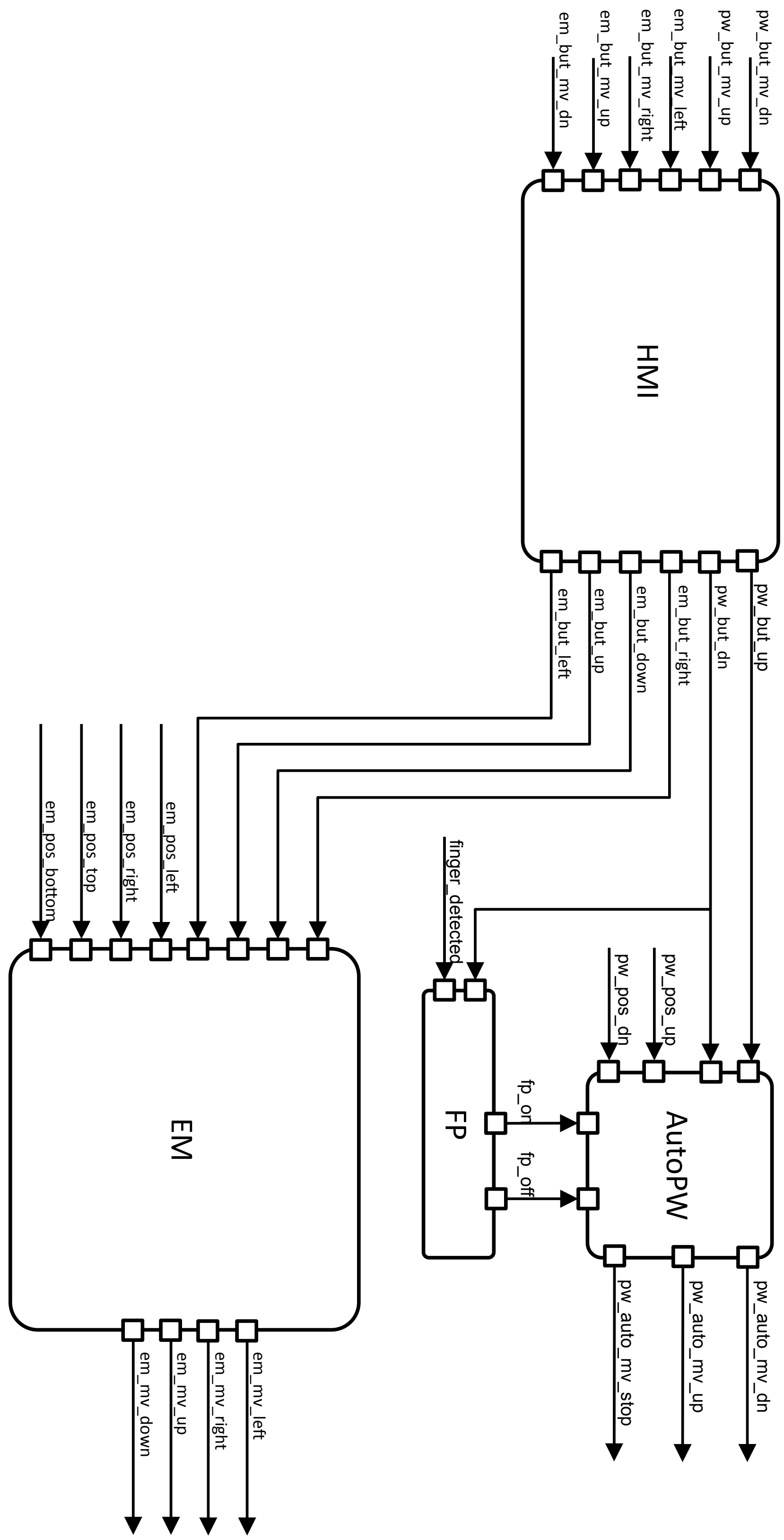

Figure 3.50.: Architecture Model Variant for Product P7

Furthermore, the architecture model comprises 29 connectors divided into 13 input connectors and seven output connectors (cf. Tab. 3.1, 3.2) as well as nine internal connectors deducible from the component interfaces. For the generation of the architecture model, we apply one architecture model delta to the core as shown in Tab. 3.4.

Architecture Model P8 The architecture model variant *P*8 is depicted in Fig. 3.51 and corresponds to product *P*8. Based on its feature configuration (cf. Tab. 2.1), the model consists of the following 11 components.

- Standard HumanMachineInterface (HMI) component
- ExteriorMirror (EMH) component variant with Heatable and LED Exterior Mirror features
- Standard LEDExteriorMirrorBottom (LED_EMB) component
- Standard LEDExteriorMirrorLeft (LED_EML) component
- Standard LEDExteriorMirrorRight (LED_EMR) component
- Standard LEDExteriorMirrorTop (LED_EMT) component
- Standard LEDExteriorMirrorHeating (LED_EMH) component
- Standard AutomaticPowerWindow (AutoPW) component
- Standard LEDAutomaticPowerWindow (LED_AutoPW) component
- Standard FingerProtection (FP) component
- Standard LEDFingerProtection (LED_FP) component

Furthermore, the architecture model comprises 64 connectors divided into 15 input connectors and 25 output connectors (cf. Tab. 3.1, 3.2) as well as 24 internal connectors deducible from the component interfaces. For the generation of the architecture model, we apply six architecture model deltas to the core as shown in Tab. 3.4.

Architecture Model P9 The architecture model variant *P*9 is shown in Fig. 3.52 and corresponds to product *P*9. Based on its feature configuration (cf. Tab. 2.2), the model consists of the following 19 components.

- HumanMachineInterface (HMI) component variant with Alarm System and LED Alarm System features
- ExteriorMirror (EMH) component variant with Heatable and LED Exterior Mirror feature
- Standard LEDExteriorMirrorBottom (LED_EMB) component
- Standard LEDExteriorMirrorLeft (LED_EML) component

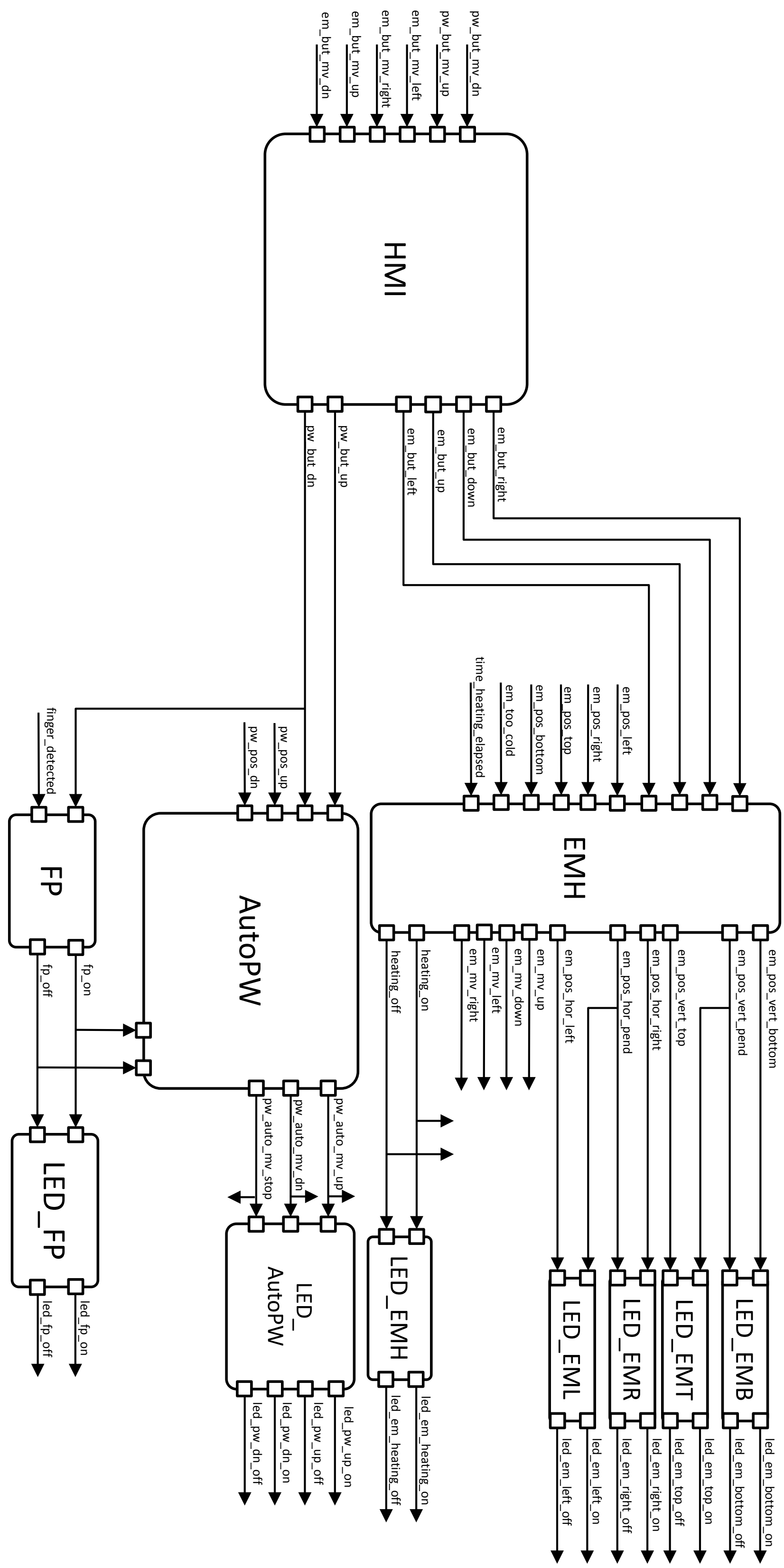

Figure 3.51.: Architecture Model Variant for Product P8

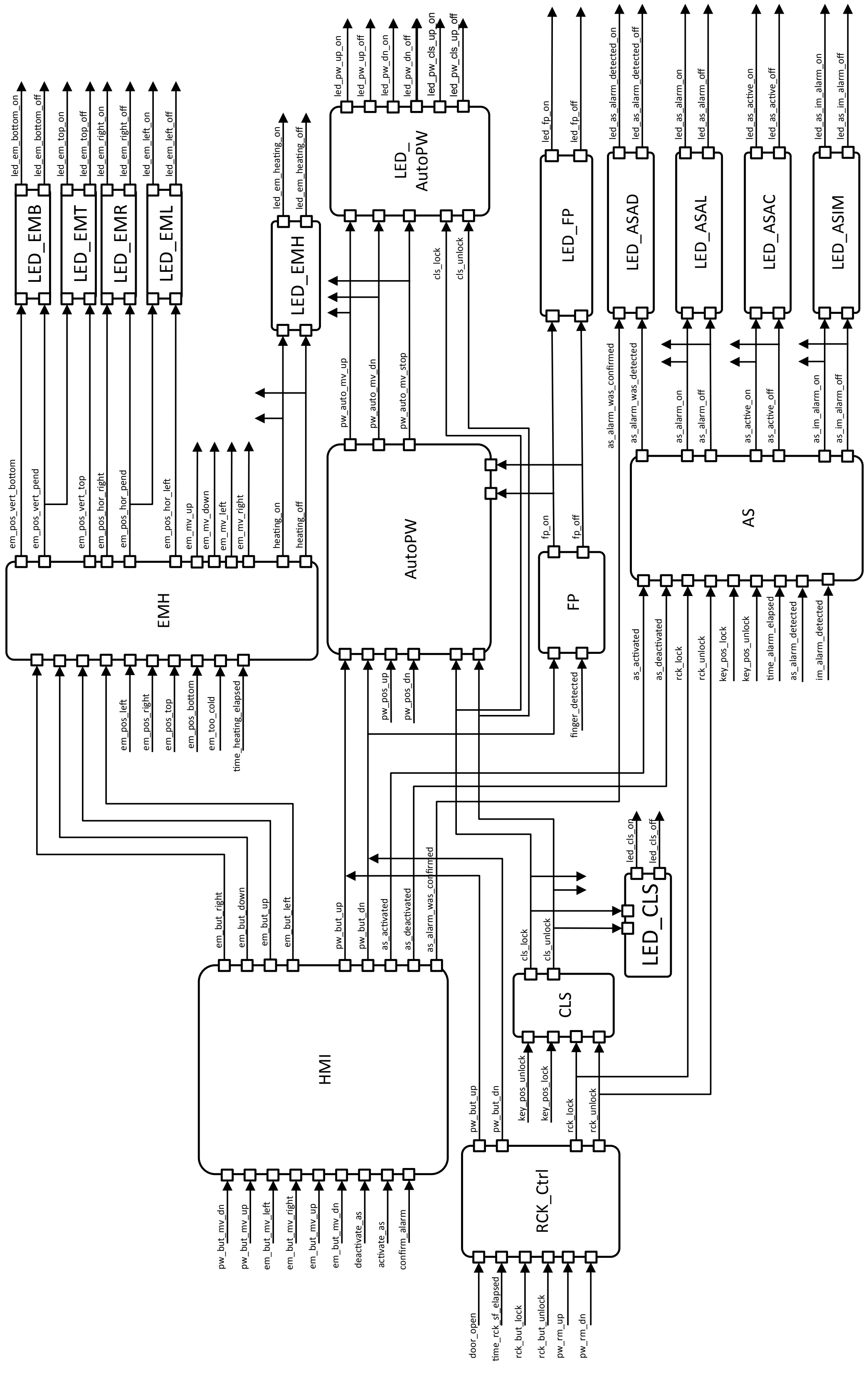

Figure 3.52.: Architecture Model Variant for Product P9

- Standard LEDExteriorMirrorRight (LED_EMR) component
- Standard LEDExteriorMirrorTop (LED_EMT) component
- Standard LEDExteriorMirrorHeating (LED_EMH) component
- AutomaticPowerWindow (AutoPW) component variant with Central Locking System feature
- Standard LEDAutomaticPowerWindow (LED_AutoPW) component
- RemoteControlKeyController (RCK_Ctrl) component variant with Control Automatic Power Window and Safety Function features
- CentralLockingSystem (CLS) component variant with Remote Control Key feature
- Standard LEDCentralLockingSystem (LED_CLS) component
- Standard FingerProtection (FP) component
- Standard LEDFingerProtection (LED_FP) component
- AlarmSystem (AS) component variant with Control Alarm System and Interior Monitoring features
- Standard LEDAlarmSystemAlarmDetected (LED_ASAD) component
- Standard LEDAlarmSystemAlarm (LED_ASAL) component
- Standard LEDAlarmSystemActive (LED_ASAC) component
- Standard LEDAlarmSystemInteriorMonitoring (LED_ASIM) component

Furthermore, the architecture model comprises 122 connectors divided into 31 input connectors and 45 output connectors (cf. Tab. 3.1, 3.2) as well as 46 internal connectors deducible from the component interfaces. For the generation of the architecture model, we apply 16 architecture model deltas to the core as shown in Tab. 3.5.

Architecture Model P10 The architecture model variant $P10$ is depicted in Fig. 3.53 and corresponds to product $P10$. Based on its feature configuration (cf. Tab. 2.2), the model consists of the following nine components.

- Standard HumanMachineInterface (HMI) component
- ExteriorMirror (EM) component variant with LED Exterior Mirror feature
- Standard LEDExteriorMirrorBottom (LED_EMB) component
- Standard LEDExteriorMirrorRight (LED_EMR) component
- Standard LEDExteriorMirrorLeft (LED_EML) component

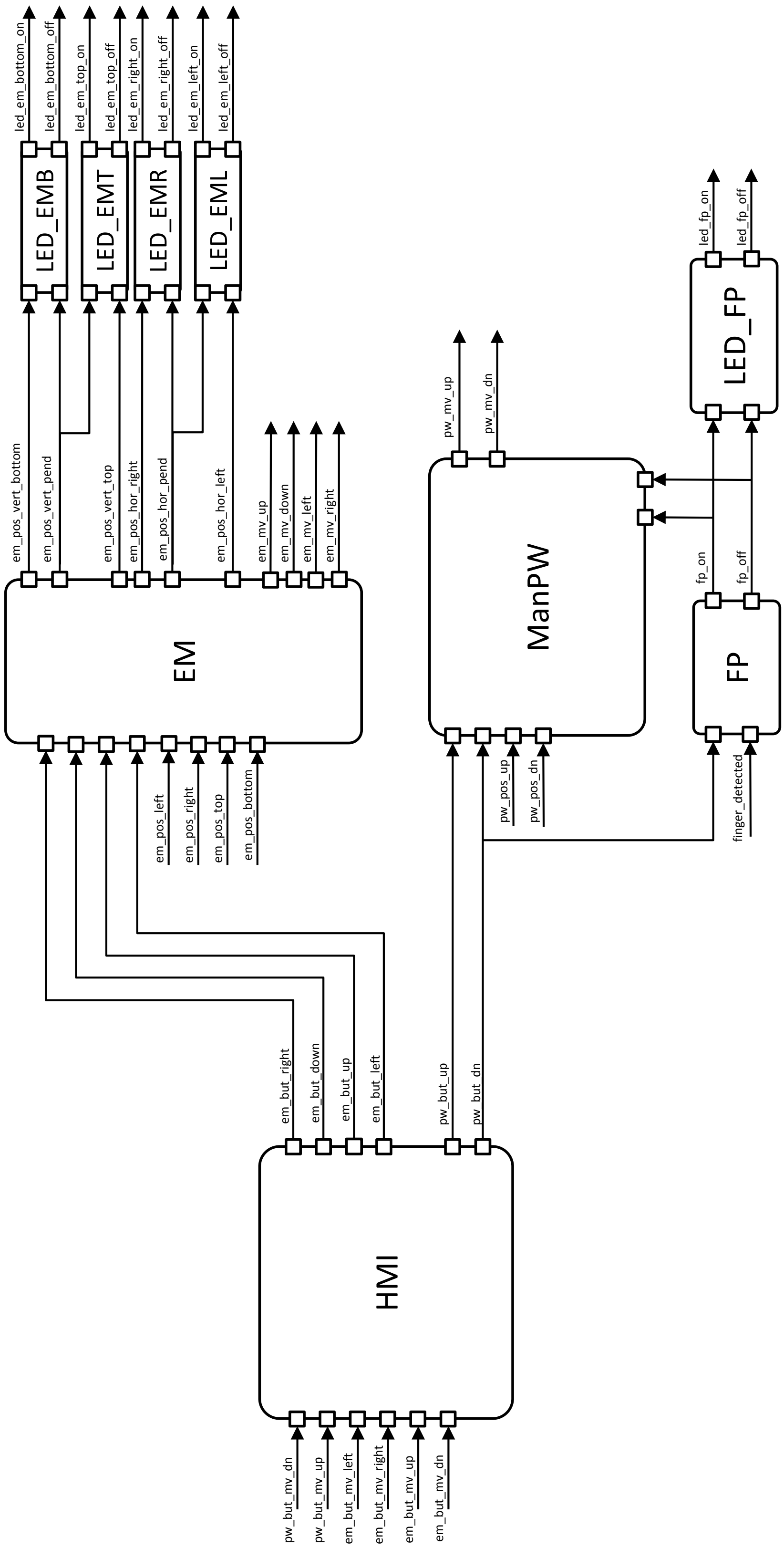

Figure 3.53.: Architecture Model Variant for Product P10

- Standard LEDExteriorMirrorTop (LED_EMT) component
- Standard ManualPowerWindow (ManPW) component
- Standard FingerProtection (FP) component
- Standard LEDFingerProtection (LED_FP) component

Furthermore, the architecture model comprises 48 connectors divided into 13 input connectors and 16 output connectors (cf. Tab. 3.1, 3.2) as well as 19 internal connectors deducible from the component interfaces. For the generation of the architecture model, we apply two architecture model deltas to the core as shown in Tab. 3.5.

Architecture Model P11 The architecture model variant $P11$ is shown in Fig. 3.54 and corresponds to product $P11$. Based on its feature configuration (cf. Tab. 2.2), the model consists of the following 14 components.

- HumanMachineInterface (HMI) component variant with Alarm System and LED Power Window features
- Standard ExteriorMirror (EM) component
- RemoteControlKeyController (RCK_Ctrl) component variant with Safety Function feature
- CentralLockingSystem (CLS) component variant with Automatic Locking and Remote Control Key features
- Standard LEDCentralLockingSystem (LED_CLS) component
- ManualPowerWindow (ManPW) component variant with Central Locking System feature
- Standard LEDManualPowerWindow (LED_ManPW) component
- Standard FingerProtection (FP) component
- Standard LEDFingerProtection (LED_FP) component
- AlarmSystem (AS) component variant with Control Alarm System feature
- Standard LEDAlarmSystemAlarmDetected (LED_ASAD) component
- Standard LEDAlarmSystemAlarm (LED_ASAL) component
- Standard LEDAlarmSystemActive (LED_ASAC) component
- Standard LEDAlarmSystemInteriorMonitoring (LED_ASAC) component

Furthermore, the architecture model comprises 95 connectors divided into 31 input connectors and 32 output connectors (cf. Tab. 3.1, 3.2) as well as 32 internal connectors deducible from the component interfaces. For the generation of the architecture model, we apply 12 architecture model deltas to the core as shown in Tab. 3.5.

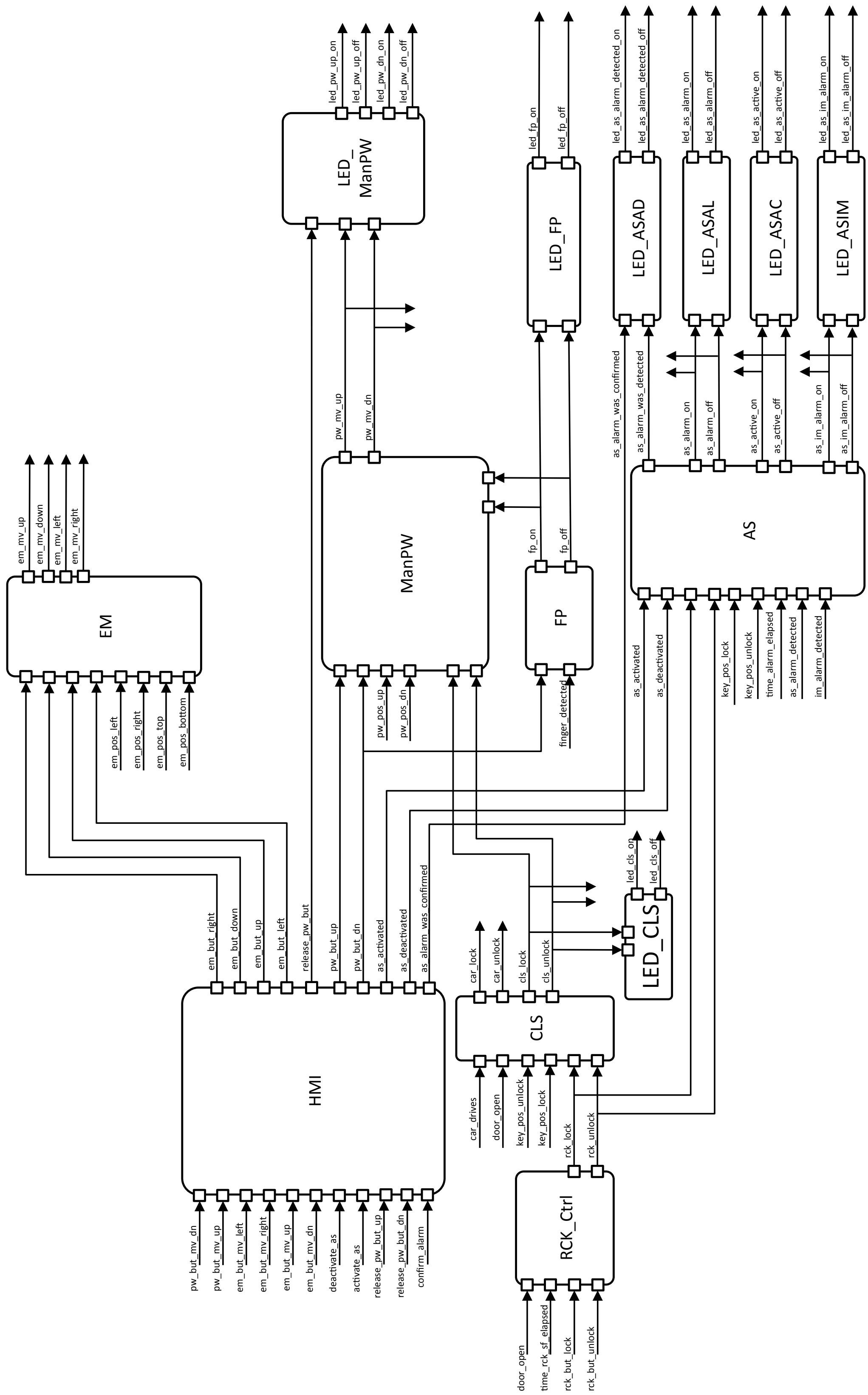

Figure 3.54.: Architecture Model Variant for Product P11

Architecture Model P12 The architecture model variant $P12$ is depicted in Fig. 3.55 and corresponds to product $P12$. Based on its feature configuration (cf. Tab. 2.2), the model consists of the following seven components.

- Standard HumanMachineInterface (HMI) component
- ExteriorMirror (EMH) component variant with Heatable feature
- AutomaticPowerWindow (AutoPW) component variant with Central Locking System feature
- Standard FingerProtection (FP) component
- RemoteControlKeyController (RCK_Ctrl) component variant with Safety Function and Control Automatic Power Window features
- CentralLockingSystem (CLS) component variant with Remote Control Key and Automatic Locking features
- Standard LEDCentralLockingSystem (LED_CLS) component

Furthermore, the architecture model comprises 57 connectors divided into 25 input connectors and 15 output connectors (cf. Tab. 3.1, 3.2) as well as 17 internal connectors deducible from the component interfaces. For the generation of the architecture model, we apply seven architecture model deltas to the core as shown in Tab. 3.5.

Architecture Model P13 The architecture model variant $P13$ is shown in Fig. 3.56 and corresponds to product $P13$. Based on its feature configuration (cf. Tab. 2.2), the model consists of the following nine components.

- Standard HumanMachineInterface (HMI) component
- ExteriorMirror (EMH) component variant with Heatable feature
- Standard LEDExteriorMirrorHeating (LED_EMH) component
- ManualPowerWindow (ManPW) component variant with Central Locking System feature
- Standard FingerProtection (FP) component
- Standard LEDFingerProtection (LED_FP) component
- CentralLockingSystem (CLS) component variant with Automatic Locking feature
- Standard LEDCentralLockingSystem (LED_CLS) component
- RemoteControlKeyController (RCK_Ctrl) component variant with Safety Function feature

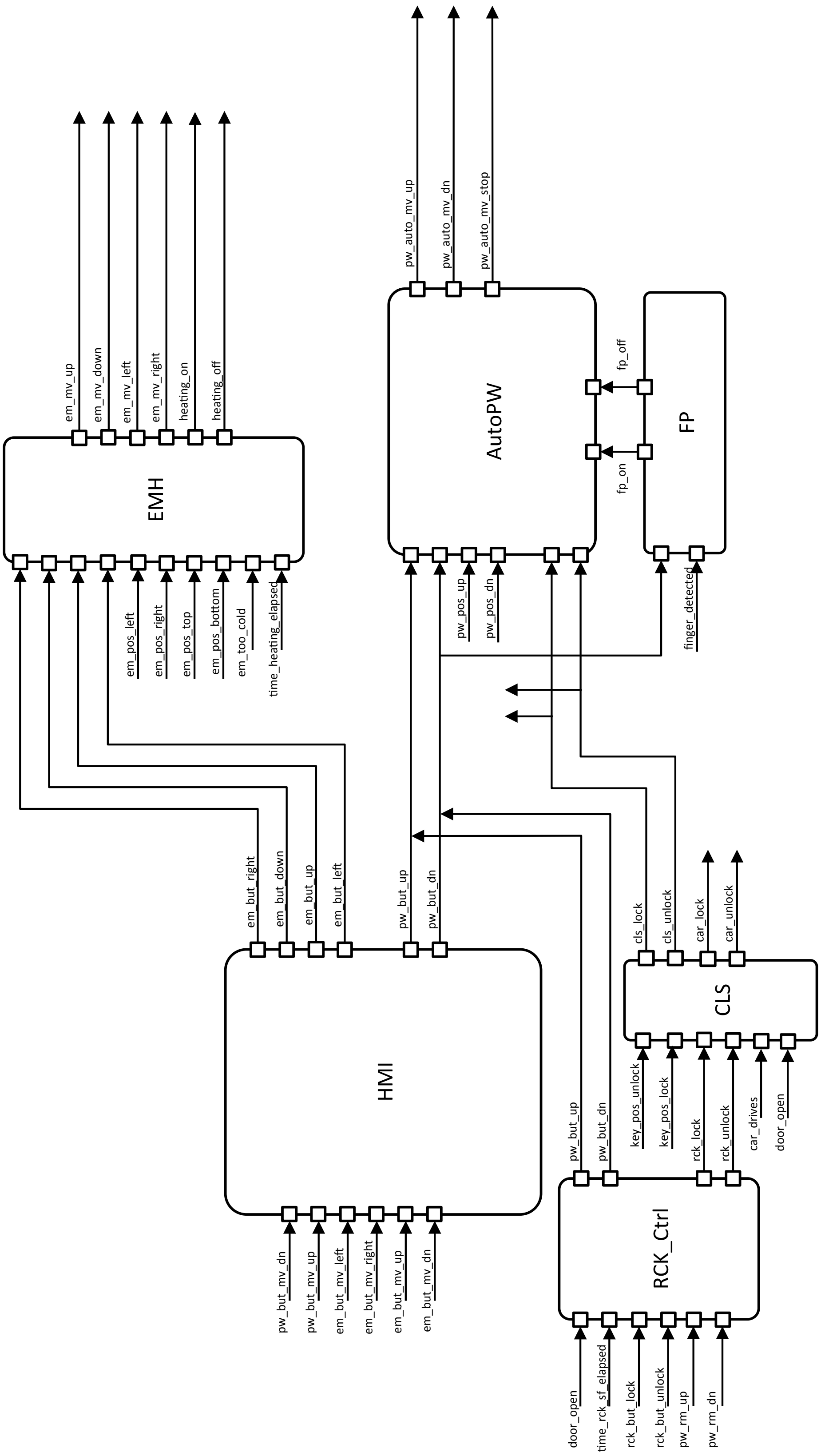

Figure 3.55.: Architecture Model Variant for Product P12

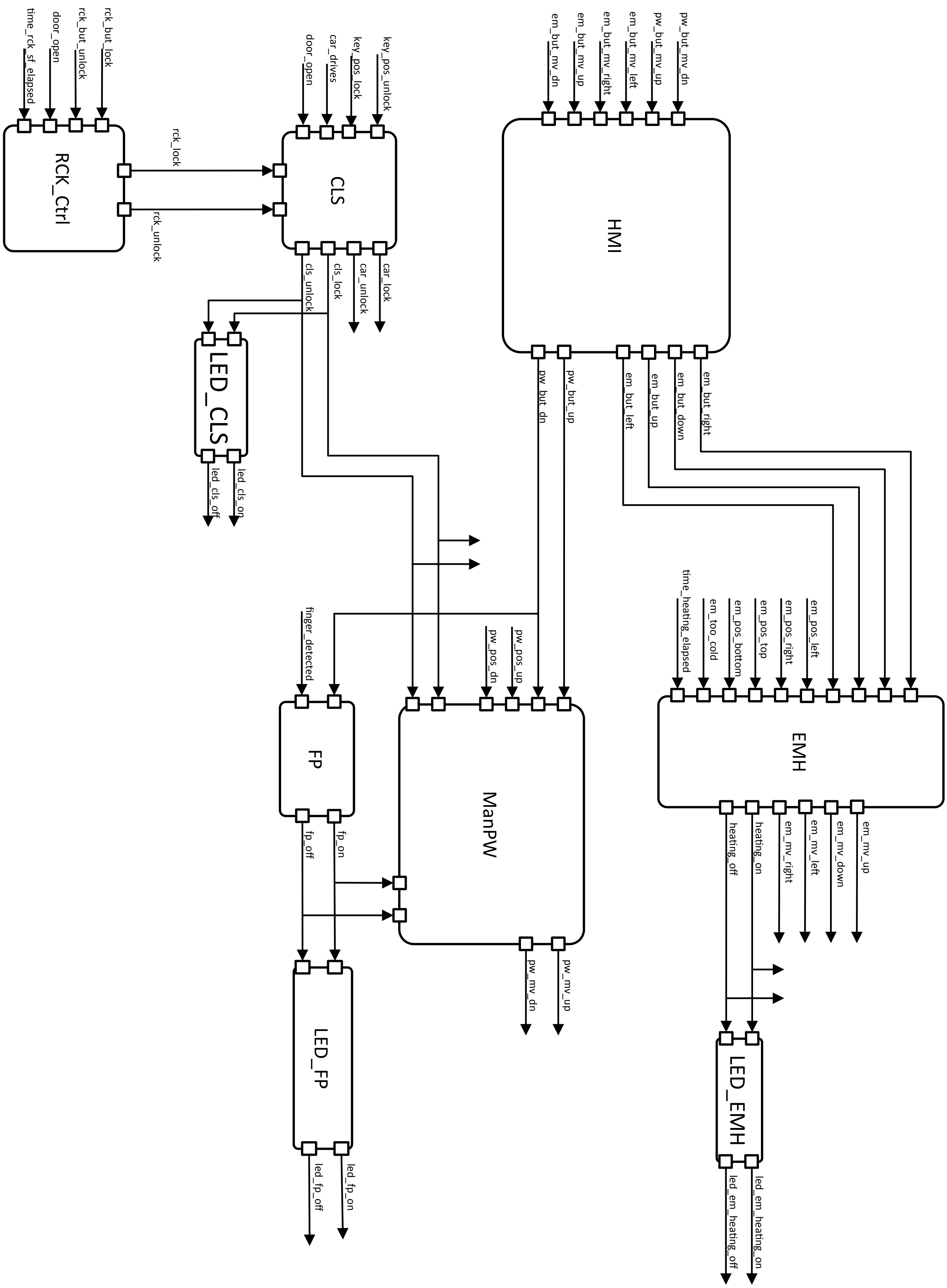

Figure 3.56.: Architecture Model Variant for Product P13

Furthermore, the architecture model comprises 60 connectors divided into 23 input connectors and 18 output connectors (cf. Tab. 3.1, 3.2) as well as 19 internal connectors deducible from the component interfaces. For the generation of the architecture model, we apply seven architecture model deltas to the core as shown in Tab. 3.5.

Architecture Model P14 The architecture model variant $P14$ is depicted in Fig. 3.57 and corresponds to product $P14$. Based on its feature configuration (cf. Tab. 2.2), the model consists of the following seven components.

- HumanMachineInterface (HMI) component variant with Alarm System feature
- ExteriorMirror (EMH) component variant with Heatable feature
- AutomaticPowerWindow (AutoPW) component variant Central Locking System feature
- RemoteControlKeyController (RCK_Ctrl) component variant with Safety Function and Control Automatic Power Window feature
- CentralLockingSystem (CLS) component variant with Automatic Locking and Remote Control Key features
- Standard FingerProtection (FP) component
- AlarmSystem (AS) component variant with Control Alarm System and Interior Mirror features

Furthermore, the architecture model comprises 70 connectors divided into 32 input connectors and 19 output connectors (cf. Tab. 3.1, 3.2) as well as 19 internal connectors deducible from the component interfaces. For the generation of the architecture model, we apply 10 architecture model deltas to the core as shown in Tab. 3.5.

Architecture Model P15 The architecture model variant $P15$ is shown in Fig. 3.58 and corresponds to product $P15$. Based on its feature configuration (cf. Tab. 2.2), the model consists of the following seven components.

- HumanMachineInterface (HMI) component variant with Alarm System
- ExteriorMirror (EMH) component variant with Heatable feature
- AutomaticPowerWindow (AutoPW) component variant with Central Locking System feature
- Standard FingerProtection (FP) component
- RemoteControlKeyController (RCK_Ctrl) component variant with Control Automatic Power Window feature
- CentralLockingSystem (CLS) component variant with Remote Control Key and Automatic Locking features

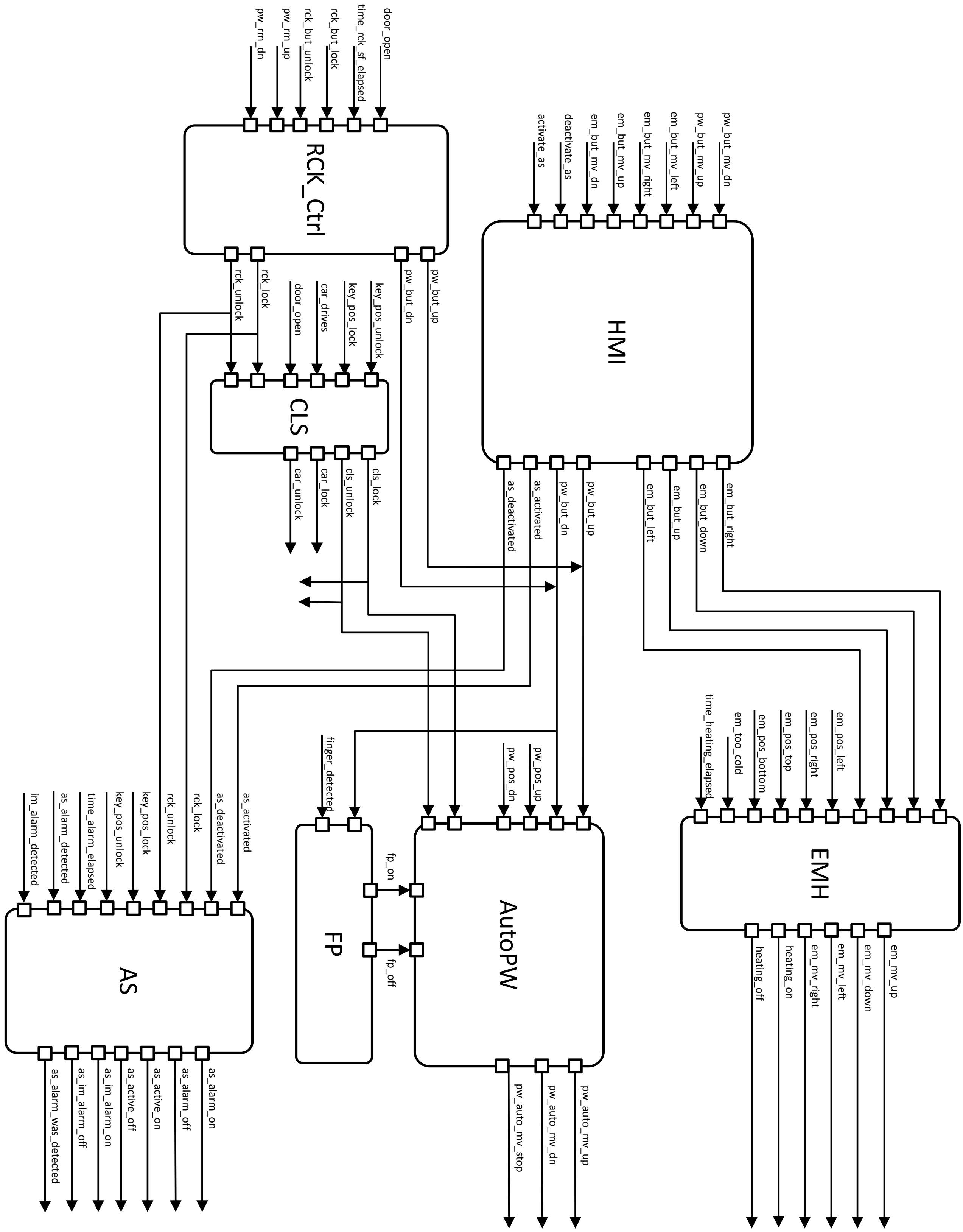

Figure 3.57.: Architecture Model Variant for Product P14

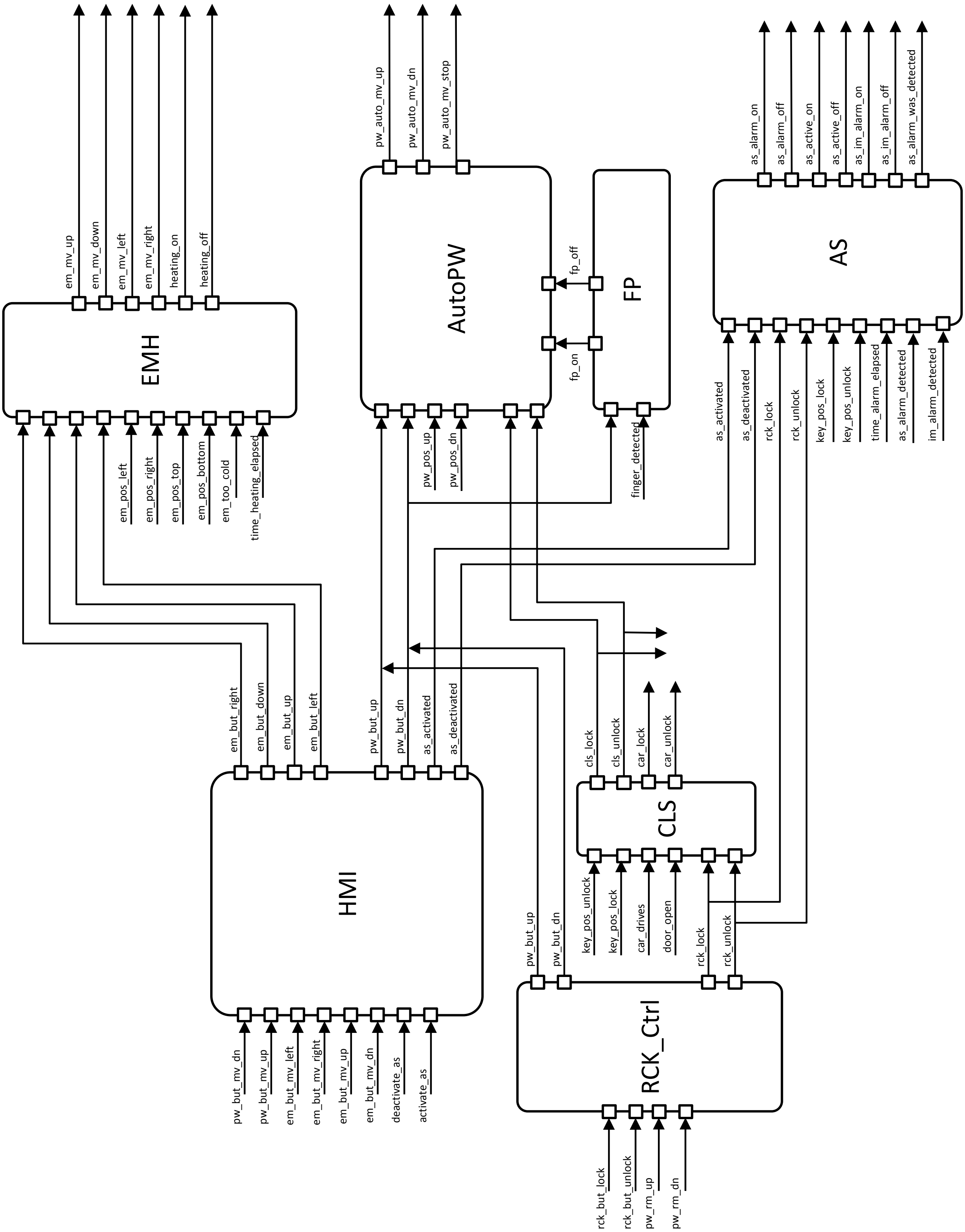

Figure 3.58.: Architecture Model Variant for Product P15

- AlarmSystem (AS) component variant with Control Alarm System and Interior Mirror features

Furthermore, the architecture model comprises 68 connectors divided into 30 input connectors and 19 output connectors (cf. Tab. 3.1, 3.2) as well as 19 internal connectors deducible from the component interfaces. For the generation of the architecture model, we apply nine architecture model deltas to the core as shown in Tab. 3.5.

Architecture Model P16 The architecture model variant $P16$ is depicted in Fig. 3.59 and corresponds to product $P16$. Based on its feature configuration (cf. Tab. 2.2), the model consists of the following 18 components.

- HumanMachineInterface (HMI) component variant with Alarm System and LED Alarm System features
- ExteriorMirror (EMH) component variant with Heatable and LED Exterior Mirror features
- Standard LEDExteriorMirrorBottom (LED_EMB) component
- Standard LEDExteriorMirrorTop (LED_EMT) component
- Standard LEDExteriorMirrorRight (LED_EMR) component
- Standard LEDExteriorMirrorLeft (LED_EML) component
- Standard LEDExteriorMirrorHeating (LED_EMH) component
- ManualPowerWindow (ManPW) component variant with Central Locking System feature
- RemoteControlKeyController (RCK_Ctrl) component variant with Safety Function feature
- CentralLockingSystem (CLS) component variant with Automatic Locking and Remote Control Key features
- Standard LEDCentralLockingSystem (LED_CLS) component
- Standard FingerProtection (FP) component
- Standard LEDFingerProtection (LED_FP) component
- AlarmSystem (AS) component variant with Control Alarm System and Interior Monitoring features
- Standard LEDAlarmSystemAlarmDetected (LED_ASAD) component
- Standard LEDAlarmSystemAlarm (LED_ASAL) component

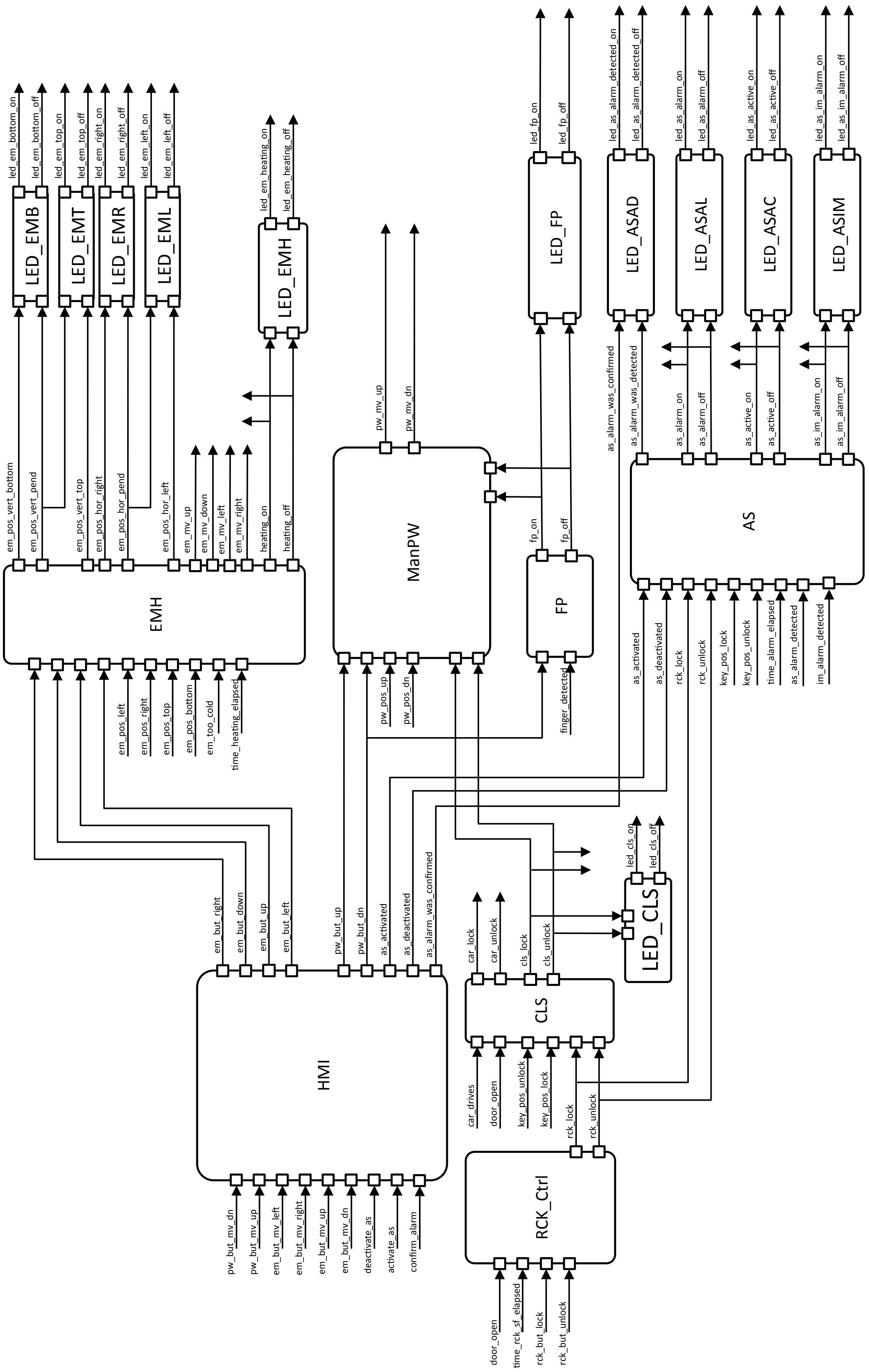

Figure 3.59.: Architecture Model Variant for Product P16

- Standard LEDAlarmSystemActive(LED_ASAC) component
- Standard LEDAlarmSystemInteriorMonitoring (LED_ASIM) component

Furthermore, the architecture model comprises 110 connectors divided into 31 input connectors and 40 output connectors (cf. Tab. 3.1, 3.2) as well as 39 internal connectors deducible from the component interfaces. For the generation of the architecture model, we apply 14 architecture model deltas to the core as shown in Tab. 3.5.

Architecture Model P17 The architecture model variant $P17$ is shown in Fig. 3.60 and corresponds to product $P17$. Based on its feature configuration (cf. Tab. 2.2), the model consists of the following four components.

- Standard HumanMachineInterface (HMI) component
- ExteriorMirror (EMH) component variant with Heatable feature
- Standard ManualPowerWindow (ManPW) component
- Standard FingerProtection (FP) component

Furthermore, the architecture model comprises 32 connectors divided into 15 input connectors and eight output connectors (cf. Tab. 3.1, 3.2) as well as nine internal connectors deducible from the component interfaces. For the generation of the architecture model, we apply one architecture model delta to the core as shown in Tab. 3.5.

For the purpose of model-based component testing of the component variants integrated in the documented architecture models, we describe in the next section the delta-oriented component state machine test models.

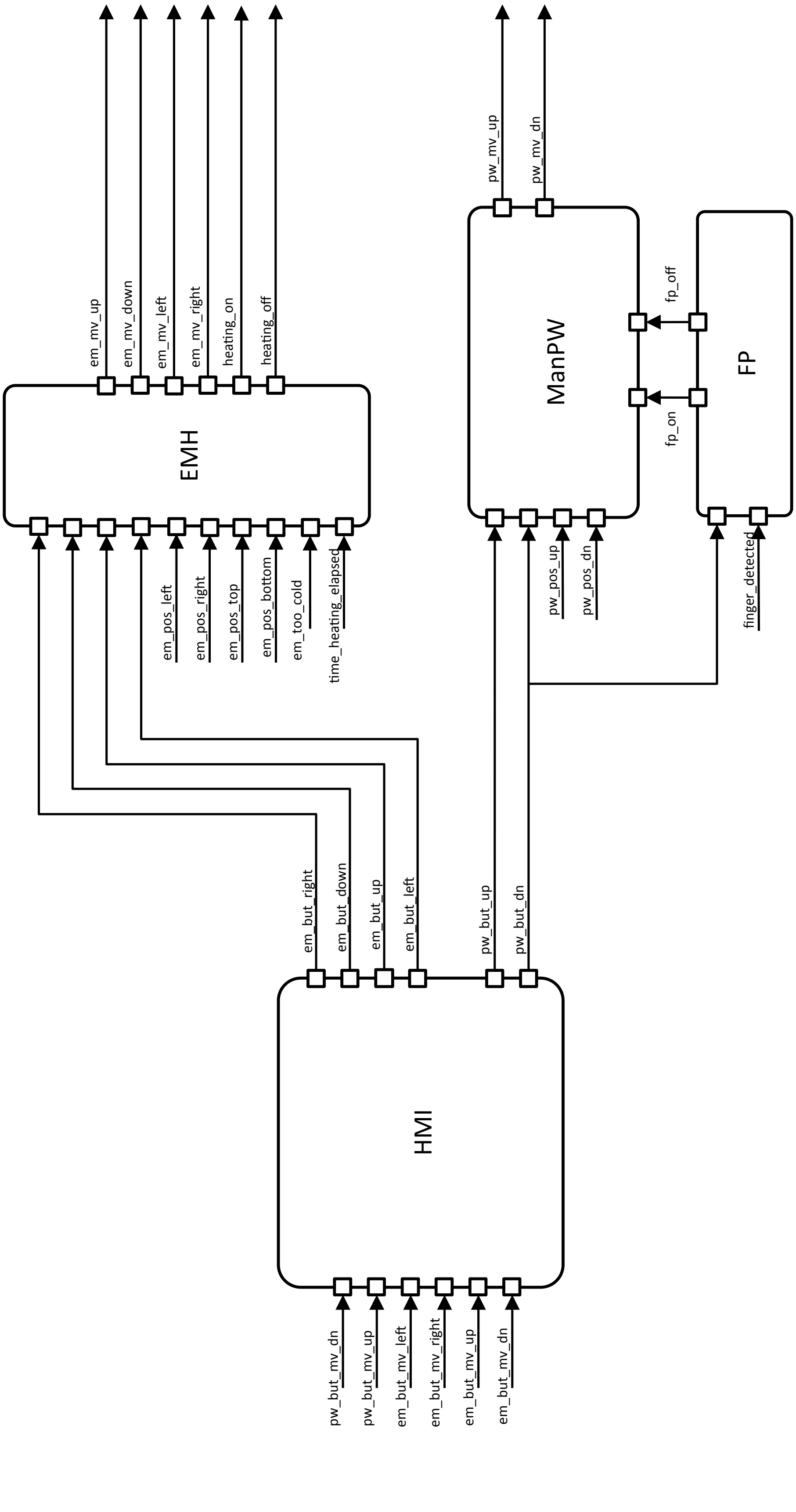

Figure 3.60.: Architecture Model Variant for Product P17

4 BCS State Machine Test Models

In this section, we document the state machine test models specifying the intended component behaviors modeled as flat i/o automata like state machines labeled either by an input (?) or by an output (!). We (1) introduce the core state machine test models, (2) describe the state machine delta models required for the generation of component test model variants, and (3) explain the resulting component state machine test model variants.

4.1. Core State Machine Test Models

Based on the description of the core components in the BCS architecture models (cf. Sect. 3), we modeled 21 corresponding core component state machine test models.

Manual Power Window The core state machine test model for the component *ManualPowerWindow* is depicted in Fig. 4.1, comprising eight states and 13 transitions. The test model specifies the behavior of the manual power window movement. The upper, lower, and pending position of the window is represented by the corresponding states *PW_up*, *PW_dn*, *PW_pending_moving_dn*, and *PW_pending_moving_up*. The initial window position is the upper position. The window starts moving downwards (*pw_mv_dn*) if the down button is pressed (*pw_but_dn)* and held. The downwards movement stops if the lower position is reached (*pw_pos_dn*). If the window stays in the lower position, the upwards movement (*pw_mv_up*) is initiated by pressing and holding the button for the upwards movement (*pw_but_up*). The upwards movement is stopped if either the window reaches the upper position (*pw_pos_up*) or the finger protection gets activated (*fp_on*), blocking the window. The window is unblocked and moves down (*pw_mv_dn*) if the finger protection is deactivated (*fp_off*). If the window stays in one of the pending positions (*PW_pending_moving_dn* and *PW_pending_moving_up*), the window is moved up or down by pressing the button for the corresponding movement.

Automatic Power Window The core state machine test model for the component *AutomaticPowerWindow* is shown in Fig. 4.2, comprising 15 states and 19 transitions. The test model specifies the behavior of the automated window movement. The upper, lower, and pending position of the window is represented by the corresponding states *PW_up*, *PW_dn*, and *PW_pend*. The initial window position is the upper position. The window starts moving downwards (*pw_auto_mv_dn*) if the down button is pressed (*pw_but_dn*). The automated downwards movement stops (*pw_auto_mv_stop*) if either the window is reaching its lower po-

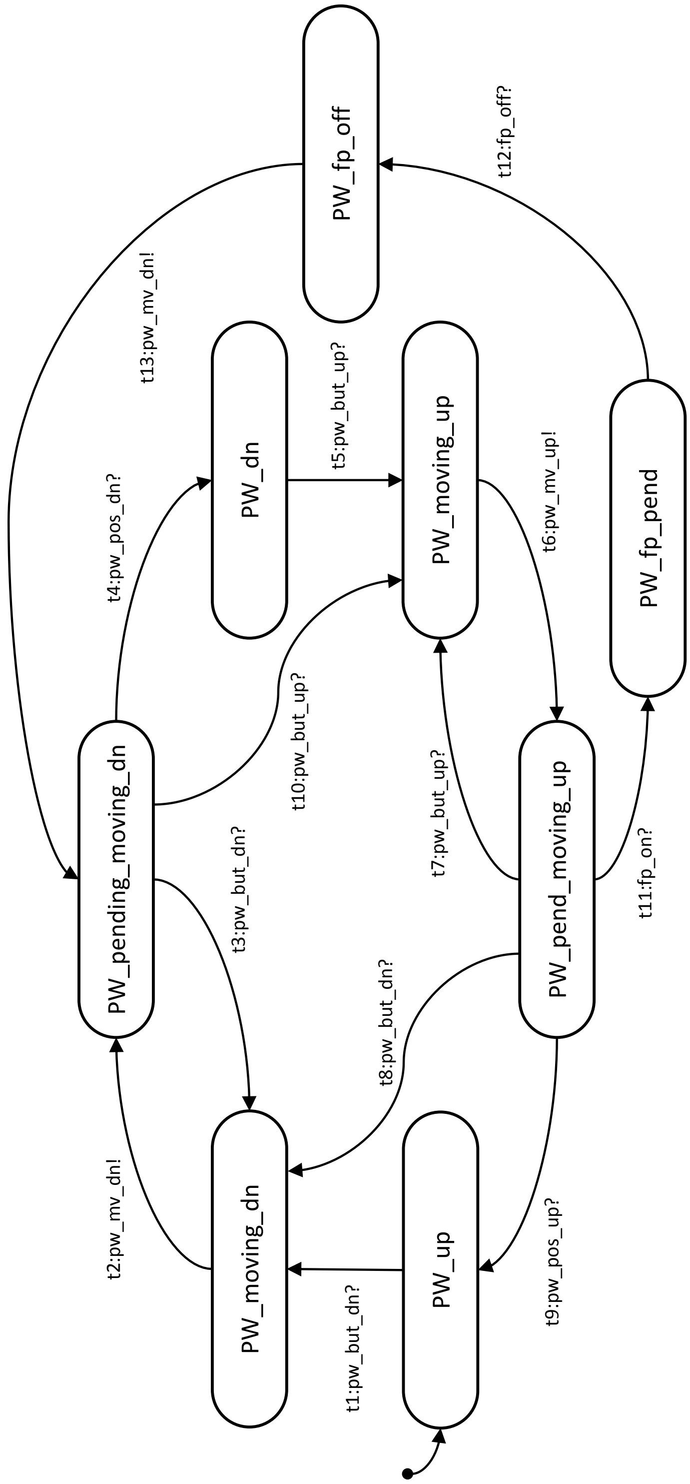

Figure 4.1.: Behavioral Specification of the Standard Manual Power Window Component

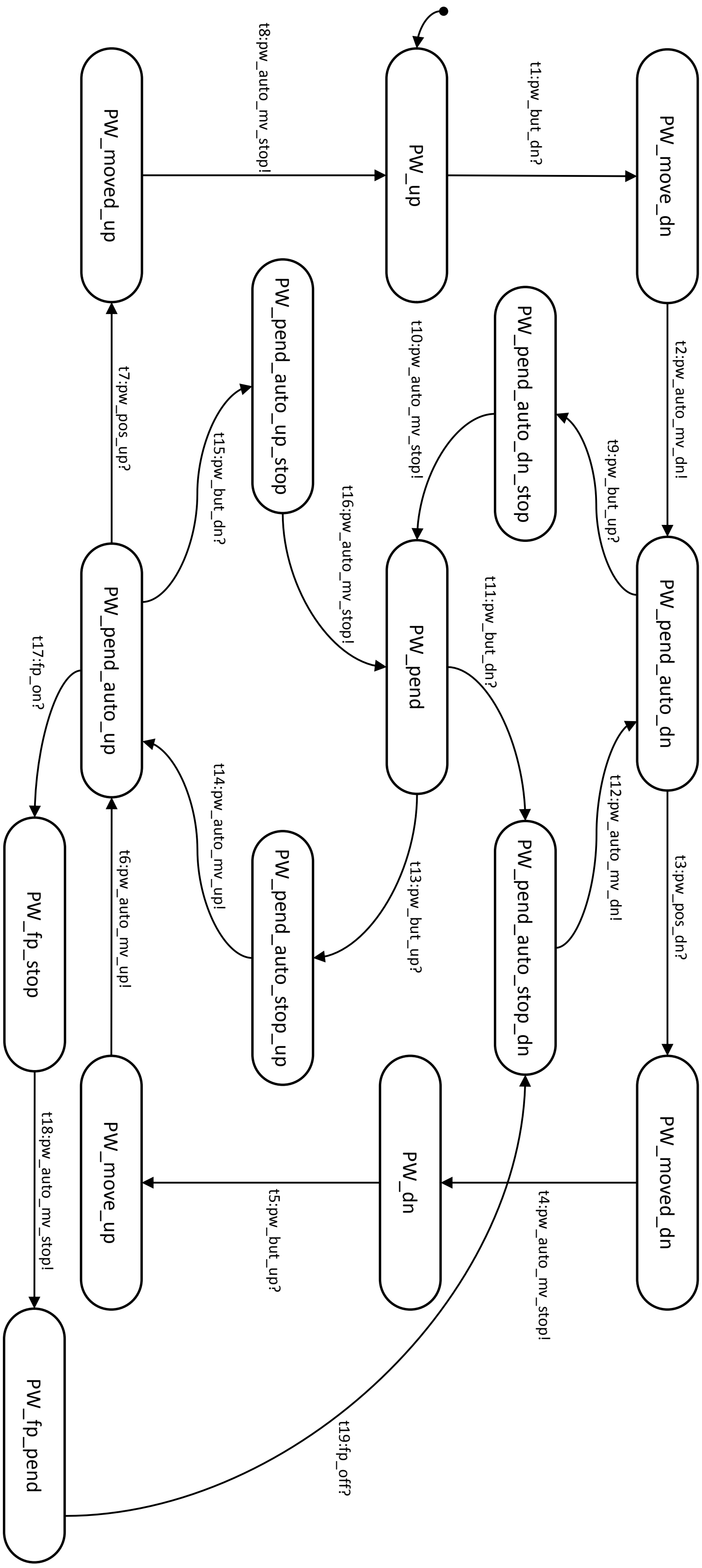

Figure 4.2.: Behavioral Specification of the Standard Automatic Power Window Component

sition (*pw_pos_dn*) or by pressing the button for the upwards movement (*pw_but_up*). If the window stays in the lower position, the automated upwards movement (*pw_auto_mv_up*) is initiated by pressing the button for the upwards movement (*pw_but_up*). The upwards movement is stopped (*pw_auto_mv_stop*) if either the window reaches the upper position (*pw_pos_up*), the button for the downwards movement is pressed (*pw_but_dn*) or the finger protection gets activated (*fp_on*), blocking the window. The window is unblocked and moves down (*pw_auto_mv_dn*) if the finger protection is deactivated (*fp_off*). If the window stays in the pending position (*PW_pend*) the window is moved upwards or downwards by pressing the button for the corresponding movement.

Finger Protection The core state machine test model for the component *FingerProtection* is depicted in Fig. 4.3, comprising four states and four transitions. The test model specifies the activation/deactivation of the finger protection. In its initial state, the finger protection is deactivated. The finger protection gets activated (*fp_on*) whenever a finger is clamped in the window (*finger_detected*). The finger protection is deactivated (*fp_off*) if the clamped finger is released due to the initiated downwards movement of the window (*pw_but_dn*).

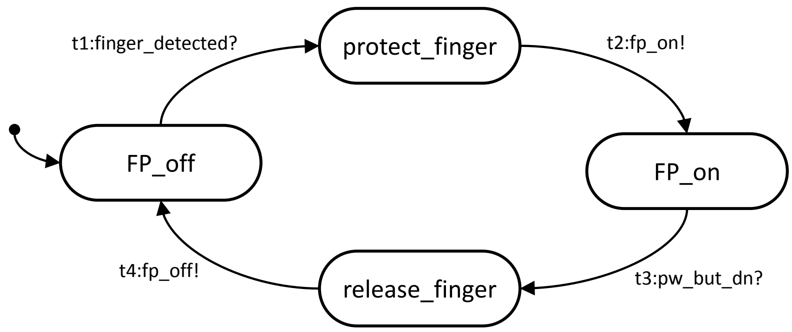

Figure 4.3.: Behavioral Specification of the Standard Finger Protection Component

Remote Control Key Controller The core state machine test model for the component *RemoteControlKeyController* is shown in Fig. 4.7, comprising three states and four transitions. The test model specifies the behavior of the remote control key controller reacting to remote signals. In its initial state, the controller remains in an idle state (*RCK_idle*), waiting for a remote signal. If the controller receives the remote signal for locking the car (*rck_but_lock*), it forwards the locking command (*rck_lock*) to the system. The same holds for the unlocking scenario, respectively.

LED Finger Protection The core state machine test model for the component *LEDFingerProtection* is shown in Fig. 4.5, comprising four states and four transitions. In its initial state (*FP_off_LED_off*), the LED is turned off. The test model specifies the turning on of the corresponding LED (*led_fp_on*) if the finger protection is activated (*fp_on*). The LED is turned off (*led_fp_off*) if the finger protection is deactivated (*fp_off*).

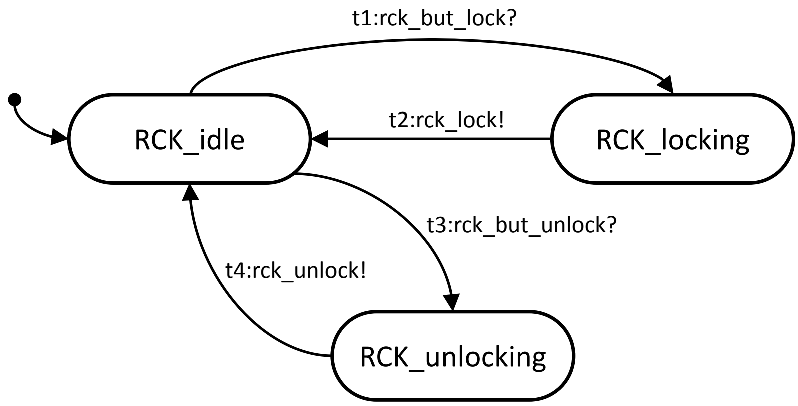

Figure 4.4.: Behavioral Specification of the Standard Remote Control Key Controller Component

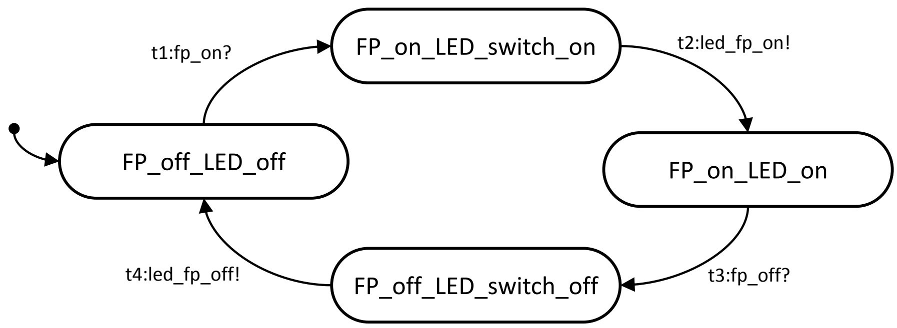

Figure 4.5.: Behavioral Specification of the Standard LED Finger Protection Component

Central Locking System The core state machine test model for the component *CentralLockingSystem* is depicted in Fig. 4.6, comprising four states and four transitions. The test model specifies the activation/deactivation of the central locking system. In its initial state, the central locking system is deactivated (*CLS_unlock*) stating that the car is unlocked. By locking the car with the car key (*key_pos_lock*), the central locking system gets activated (*cls_lock*). If the car is unlocked with the car key (*key_pos_unlock*), the central locking system gets deactivated (*cls_unlock*), respectively.

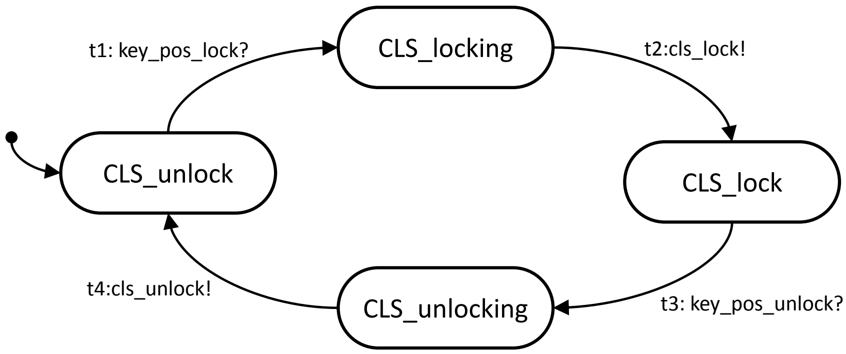

Figure 4.6.: Behavioral Specification of the Standard Central Locking System Component

Exterior Mirror The core state machine test model for the component *ExteriorMirror* is shown in Fig. 4.7, comprising 21 states and 48 transitions. The test model specifies the behavior of the exterior mirror position adjustment. The upper, upper left, upper right, lower, lower left, lower right, left, right, and pending position of the mirror is represented by the corresponding states *EM_top*, *EM_top_left*, *EM_top_right*, *EM_bottom*, *EM_bottom_left*, *EM_bottom_right*, *EM_hor_left*, *EM_hor_right*, and *EM_hor_pending*. The initial window position is the pending position (*EM_hor_pending*). Starting in the initial state, the exterior mirror moves up (*em_mv_up*), down (*em_mv_down*), left (*em_mv_left*), or right (*em_mv_right*) based on the corresponding movement command (*em_but_up*, *em_but_down*, *em_but_left*, *em_but_right*). If the mirror reaches (*em_pos_top*, *em_pos_bottom*, *em_pos_left*, *em_pos_right*) one of its end positions, it stops moving in the corresponding direction. Based on its current position, the mirror is able to move into the remaining directions until a new end position is reached.

Human Machine Interface The core state machine test model for the component *HumanMachineInterface* is shown in Fig. 4.8, comprising seven states and 12 transitions. The test model specifies the behavior of the human machine interface reacting to the interaction with the driver. In its initial state, the human machine interface remains in an idle state (*Controller*) waiting for a signal representing the driver interaction. If the human machine interface receives the signal for moving downwards the window (*pw_but_mv_dn*), it forwards the window movement command (*pw_but_dn*) to the system. The same situation occurs for the upwards movement of the window (*pw_but_mv_up*, *pw_but_up*). If one of

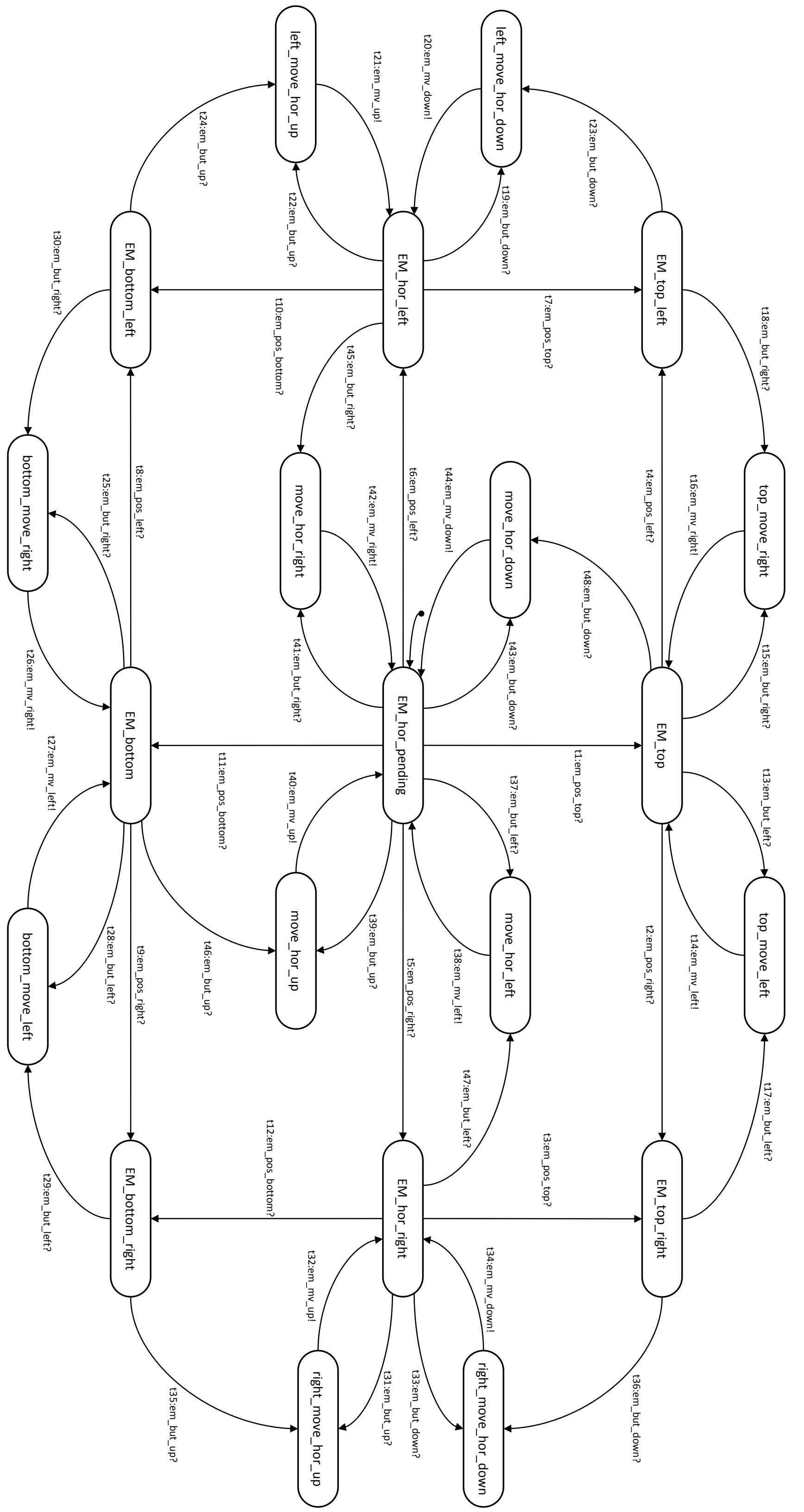

Figure 4.7.: Behavioral Specification of the Standard Exterior Mirror Component

the signals for adjusting the exterior mirror (*em_but_mv_up*, *em_but_mv_dn*, *em_but_mv_left*, *em_but_mv_right*) is received, it forwards the corresponding adjustment command (*em_but_up*, *em_but_dn*, *em_but_left*, *em_but_right*) to the system.

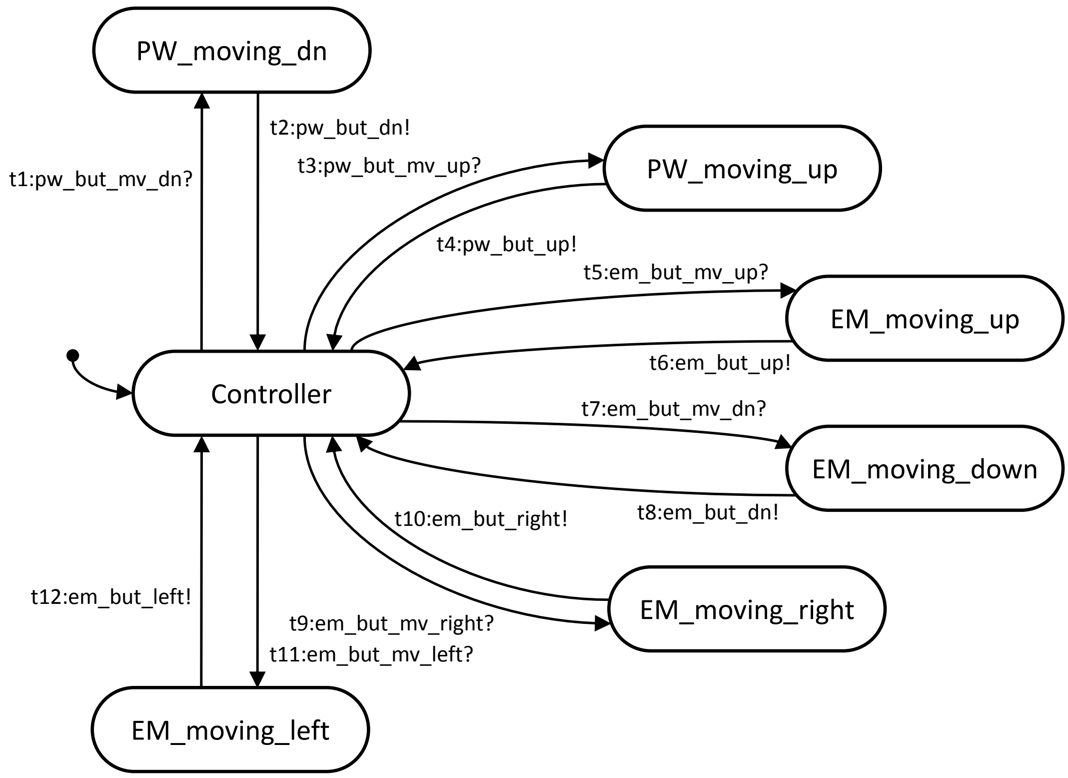

Figure 4.8.: Behavioral Specification of the Standard Human Machine Interface Component

LED Central Locking System The core state machine test model for the component *LEDCentralLockingSystem* is depicted in Fig. 4.9, comprising four states and four transitions. In its initial state (*CLS_LED_off*), the LED is turned off. The test model specifies the turning on of the respective LED (*led_cls_on*) if the central locking system is activated, i.e., the car is locked (*cls_lock*). The LED is turned off (*led_cls_off*) if the central locking system is deactivated by unlocking the car (*cls_unlock*).

LED Manual Power Window The core state machine test model for the component *LEDManualPowerWindow* is depicted in Fig. 4.10, comprising seven states and eight transitions. In its initial state (*LED_ManPW_off*), both LEDs are turned off. The test model specifies the turning on of the LED for the downwards movement (*led_pw_dn_on*), if the manual power window moves down (*pw_mv_dn*). The LED is turned off (*led_pw_dn_off*) if the window stops moving based on the released button (*release_pw_but*). A similar behavior is specified for the LED for the upwards movement.

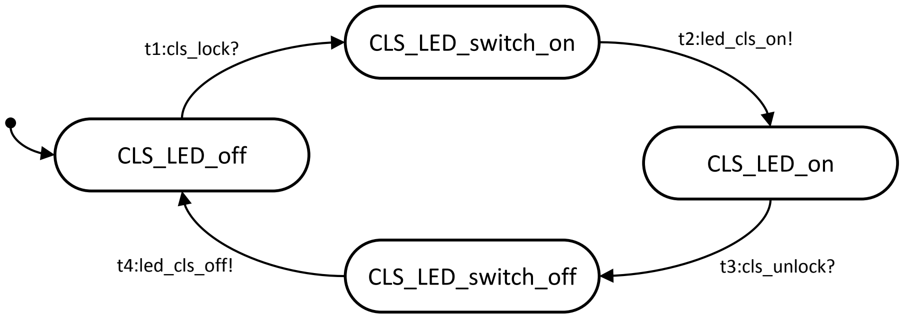

Figure 4.9.: Behavioral Specification of the Standard LED Central Locking System Component

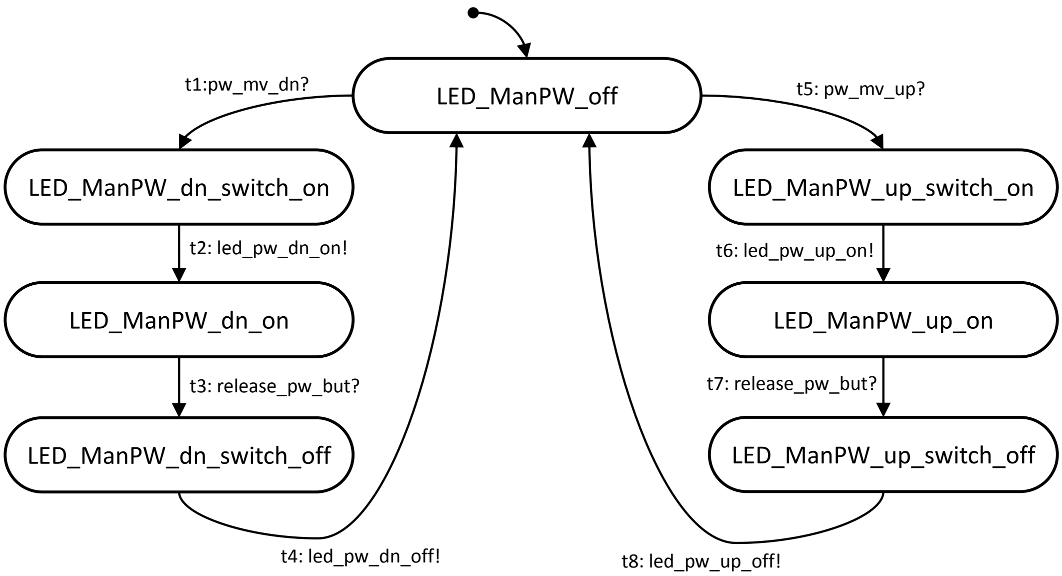

Figure 4.10.: Behavioral Specification of the Standard LED Manual Power Window Component

LED Automatic Power Window The core state machine test model for the component *LEDAutomaticPowerWindow* is shown in Fig. 4.11, comprising seven states and eight transitions. In its initial state (*LED_AutoPW_off*), both LEDs are turned off. The test model specifies the turning on of the LED for the downwards movement (*led_pw_dn_on*) if the automatic power window moves down (*pw_auto_mv_dn*). The LED is turned off (*led_pw_dn_off*) if the window stops its automated movement (*pw_auto_mv_stop*). A similar behavior is specified for the LED for the upwards movement.

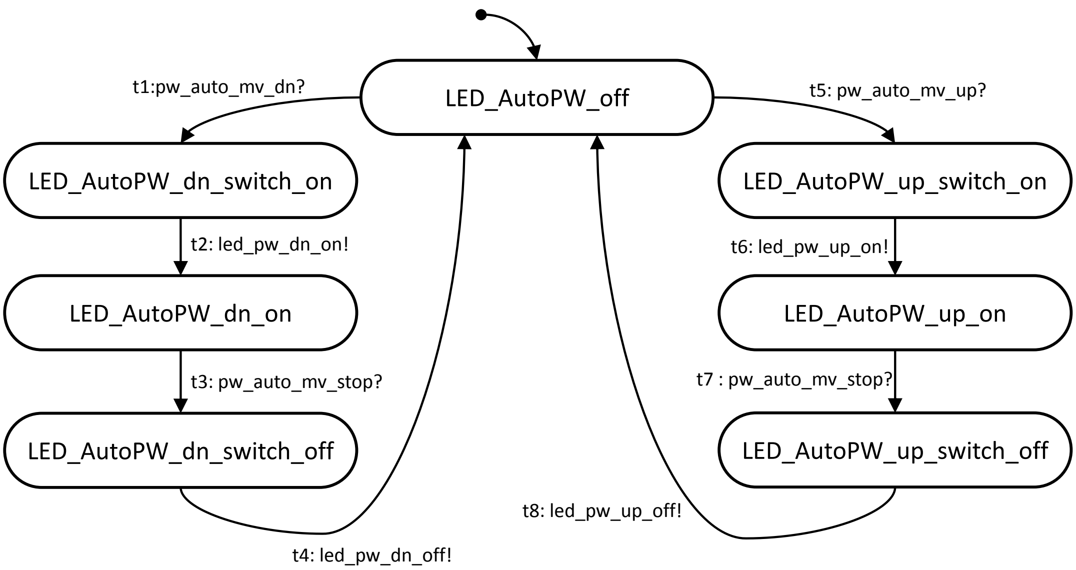

Figure 4.11.: Behavioral Specification of the Standard LED Automatic Power Window Component

LED Exterior Mirror Top The core state machine test model for the component *LEDExteriorMirrorTop* is depicted in Fig. 4.12, comprising four states and four transitions. In its initial state (*EM_LED_top_off*), the LED is turned off. The test model specifies the turning on of the corresponding LED (*led_em_top_on*) if the exterior mirror reaches the upper position (*em_pos_vert_top*). The LED is turned off (*led_em_top_off*) if the exterior mirror leaves the upper position, i.e., pending between the upper and lower position (*em_pos_vert_pend*).

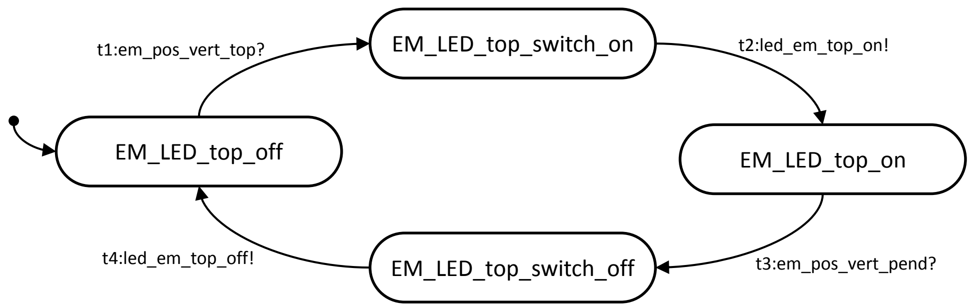

Figure 4.12.: Behavioral Specification of the Standard LED Exterior Mirror Top Component

LED Exterior Mirror Left The core state machine test model for the component *LEDExteriorMirrorLeft* is shown in Fig. 4.13, comprising four states and four transitions. In its initial state (*EM_LED_left_off*), the LED is turned off. The test model specifies the turning on of the corresponding LED (*led_em_left_on*) if the exterior mirror reaches the left-most position (*em_pos_hor_left*). The LED is turned off (*led_em_left_off*) if the exterior mirror leaves the left-most position, i.e., pending between the left-most and right-most position (*em_pos_hor_pend*).

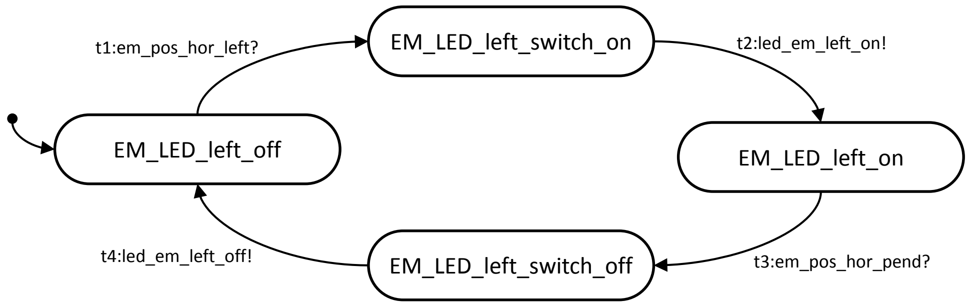

Figure 4.13.: Behavioral Specification of the Standard LED Exteriror Mirror Left Component

LED Exterior Mirror Bottom The core state machine test model for the component *LEDExteriorMirrorBottom* is depicted in Fig. 4.14, comprising four states and four transitions. In its initial state (*EM_LED_bottom_off*), the LED is turned off. The test model specifies the turning on of the corresponding LED (*led_em_bottom_on*) if the exterior mirror reaches the lower position (*em_pos_vert_bottom*). The LED is turned off (*led_em_bottom_off*) if the exterior mirror leaves the lower position, i.e., pending between the lower and upper position (*em_pos_vert_pend*).

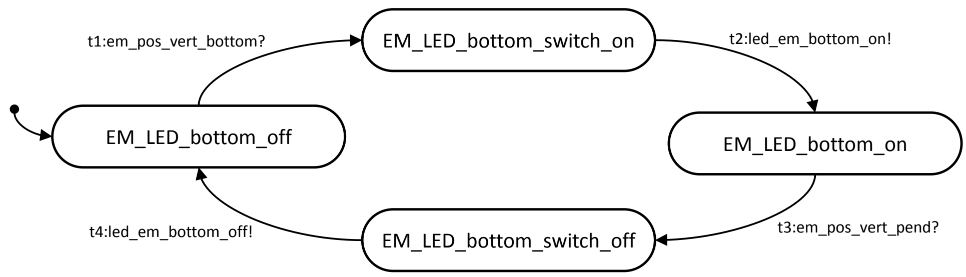

Figure 4.14.: Behavioral Specification of the Standard LED Exterior Mirror Bottom Component

LED Exterior Mirror Right The core state machine test model for the component *LEDExteriorMirrorRight* is shown in Fig. 4.15, comprising four states and four transitions. In its initial state (*EM_LED_right_off*), the LED is turned off. The test model specifies the turning on of the corresponding LED (*led_em_right_on*) if the exterior mirror reaches the right-most position (*em_pos_hor_right*). The LED is turned off (*led_em_right_off*) if the exterior mirror leaves the right-most position, i.e., pending between the right-most and left-most position (*em_pos_hor_pend*).

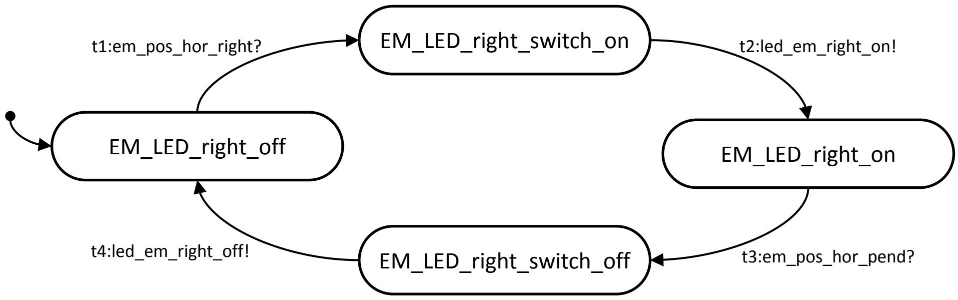

Figure 4.15.: Behavioral Specification of the Standard LED Exterior Mirror Right Component

LED Exterior Mirror Heating The core state machine test model for the component *LEDExteriorMirrorHeating* is depicted in Fig. 4.16, comprising four states and four transitions. In its initial state (*EM_heating_LED_off*), the LED is turned off. The test model specifies the turning on of the corresponding LED (*led_em_heating_on*) if the heater of the exterior mirror is activated (*heating_on*). The LED is turned off (*led_em_heating_off*) if the heater of the exterior mirror is deactivated (*heating_off*).

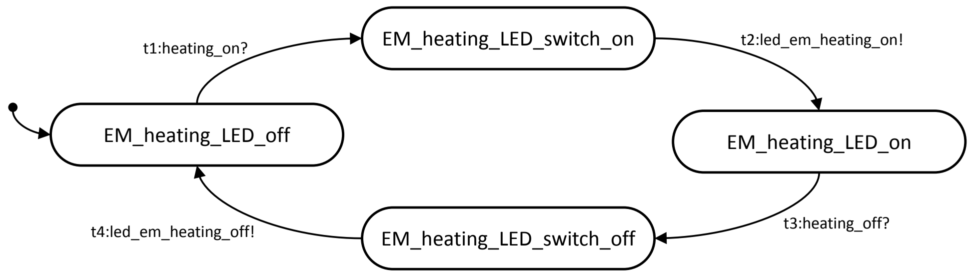

Figure 4.16.: Behavioral Specification of the Standard LED Exterior Mirror Heating Component

LED Alarm System Active The core state machine test model for the component *LEDAlarmSystemActive* is depicted in Fig. 4.17, comprising four states and four transitions. In its initial state (*AS_active_LED_off*), the LED is turned off. The test model specifies the turning on of the corresponding LED (*led_as_active_on*) if the alarm monitoring of the alarm system is activated (*as_active_on*). The LED is turned off (*led_as_active_off*) if the alarm monitoring is deactivated (*as_active_off*).

LED Alarm System Alarm The core state machine test model for the component *LEDAlarmSystemAlarm* is shown in Fig. 4.18, comprising four states and four transitions. In its initial state (*AS_alarm_LED_off*), the LED is turned off. The test model specifies the turning on of the corresponding LED (*led_as_alarm_on*) if the alarm is triggered (*as_alarm_on*). The LED is turned off (*led_as_alarm_off*) if the alarm is stopped (*as_alarm_off*).

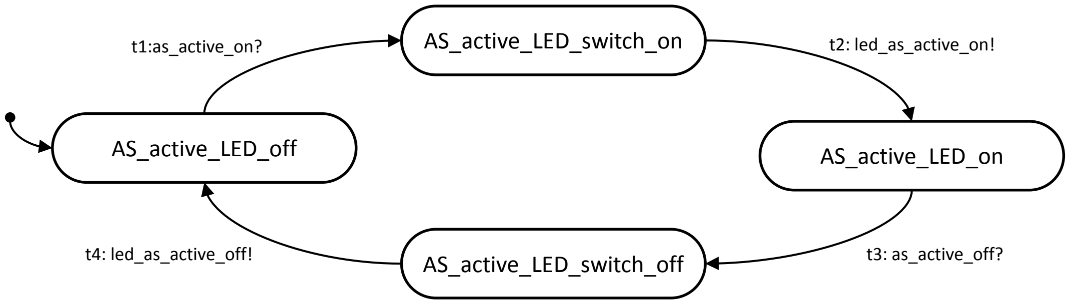

Figure 4.17.: Behavioral Specification of the Standard LED Alarm System Active Component

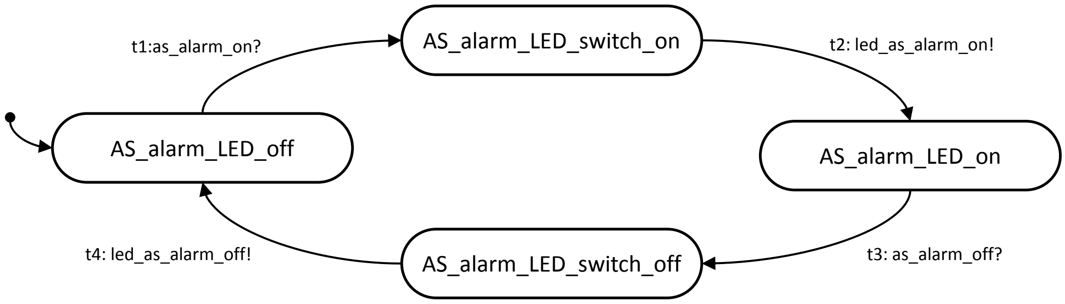

Figure 4.18.: Behavioral Specification of the Standard LED Alarm System Alarm Component

LED Alarm System Alarm Detected The core state machine test model for the component *LEDAlarmSystemAlarmDetected* is depicted in Fig. 4.19, comprising four states and four transitions. In its initial state (*AS_alarm_detected_LED_off*), the LED is turned off. The test model specifies the turning on of the corresponding LED (*led_as_alarm_detected_on*) if the alarm is stopped based on the elapsed alarm time sending a silent alarm (*as_alarm_was_detected*). The LED is turned off (*led_as_alarm_detected_off*) if the detected alarm is confirmed by the driver (*as_*
alarm_was_confirmed).

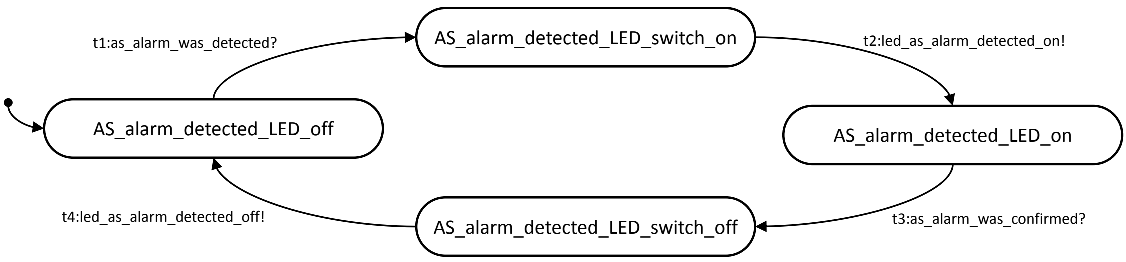

Figure 4.19.: Behavioral Specification of the Standard LED Alarm System Alarm Detected Component

LED Alarm System Interior Monitoring The core state machine test model for the component *LEDAlarmSystemInteriorMonitoring* is shown in Fig. 4.20, comprising four states and four transitions. In its initial state (*AS_im_alarm_LED_off*), the LED is turned off. The test model specifies the turning on of the corresponding LED (*led_as_im_alarm_on*) if the interior alarm is triggered (*as_im_alarm_on*). The LED is turned off (*led_as_im_alarm_off*) if the interior alarm is stopped (*as_im_alarm_off*).

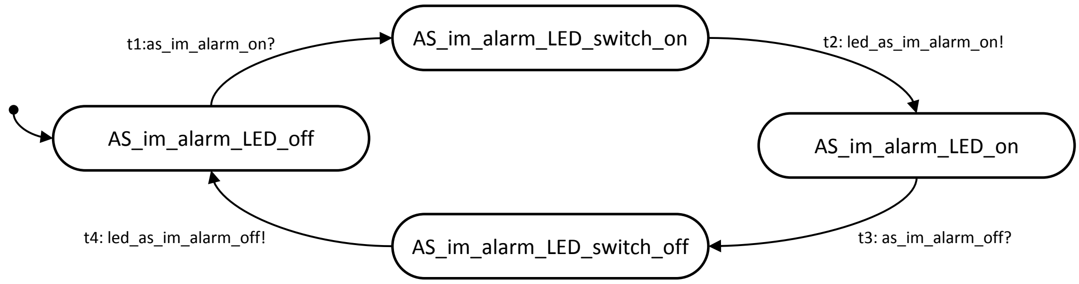

Figure 4.20.: Behavioral Specification of the Standard LED Alarm System Interior Monitoring Alarm Component

Alarm System The core state machine test model for the component *AlarmSystem* is depicted in Fig. 4.21, comprising 10 states and 13 transitions. The test model specifies the behavior of the activation/deactivation of the alarm system as well as the enabling/disabling of the alarm monitoring. In its initial state (*AS_activated_off*) the alarm system is activated and the monitoring is disabled. The alarm system can be deactivated (*as_deactivated*) and re-activated again (*as_deactivated*). The alarm monitoring of the alarm system is enabled (*as_active_on*) if the car is locked by using the car key (*key_pos_lock*). The active system is disabled (*as_active_off*) if the car is unlocked (*key_pos_unlock*). If the alarm monitoring is enabled (*AS_on*) and an alarm is detected (*as_alarm_detected*), the alarm is triggered (*as_alarm_on*). The triggered alarm is stopped (*as_alarm_off*) if either the car is unlocked (*key_pos_unlock*) or the alarm time elapsed (*time_alarm_elapsed*) sending a silent alarm (*alarm_was_detected*).

Based on the core state machine test models, we describe the state machine delta models specifying the changes for transforming the different cores into the test model variants documented in Sect. 4.3.

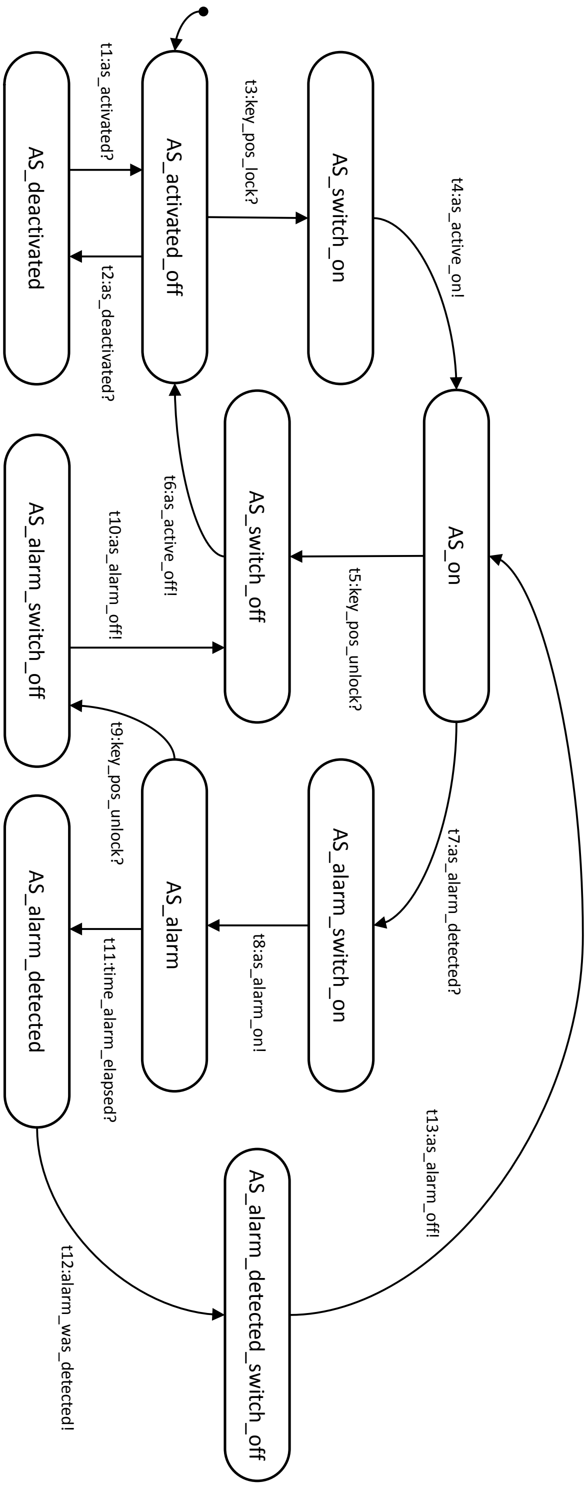

Figure 4.21.: Behavioral Specification of the Standard Alarm System Component

4.2. State Machine Delta Models

In this section, we present the different state machine delta models defined for the transformation of the different core test models. In the BCS SPL case study, there are some component state machine test models where the core specifies the complete behavior, i.e., there is no variant changing the behavior of the core. Therefore, we define for the following components no state machine delta models.

- Finger Protection
- LED Alarm System Active
- LED Alarm System Alarm
- LED Alarm System Alarm Detected
- LED Alarm System Interior Monitoring
- LED Central Locking System
- LED Exterior Mirror Top
- LED Exterior Mirror Bottom
- LED Exterior Mirror Left
- LED Exterior Mirror Right
- LED Exterior Mirror Heating
- LED Finger Protection
- LED Manual Power Window

In total, we modeled 15 state machine deltas as follows, where each delta is mapped to its corresponding component. Please note that an *addition* of an element is represented by a **+** (plus) and a *removal* of an element is illustrated by a **-** (minus) in the graphical representation of the corresponding element. Furthermore, existing elements, i.e., elements of the corresponding core model as well as elements added by previously applied deltas, are represented by dashed borderlines.

Manual Power Window We define one delta applied to the core if the features *Manual Power Window* and *Central Locking system* are selected for a product configuration. The delta *DAddManPWCLS* shown in Fig. 4.22 transforms the corresponding core such that the window movement of the manual power window is blocked by an active central locking system and unblocked based on the deactivation of the central locking system. Therefore, we add five states and 14 transitions to integrate the new behavior into the core state machine test model.

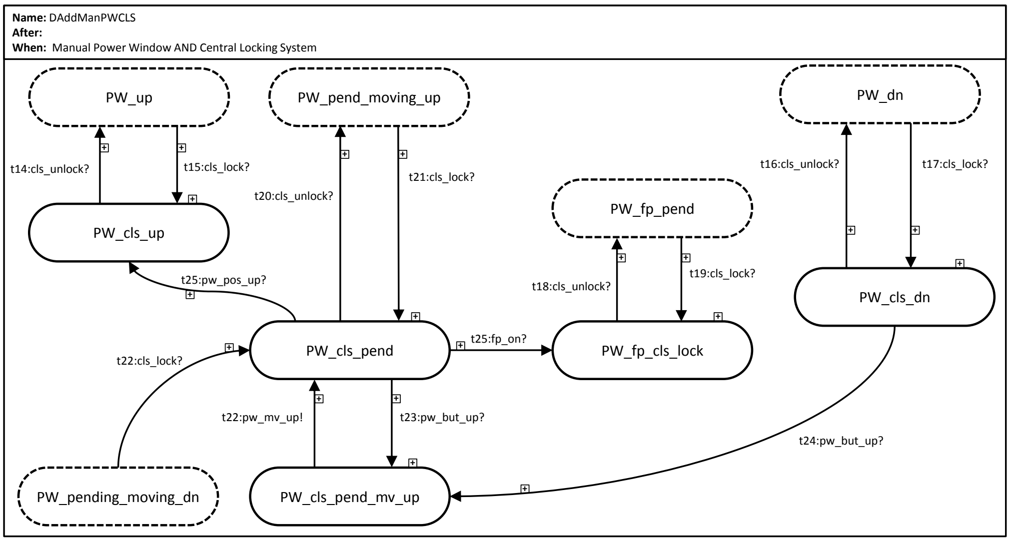

Figure 4.22.: State Machine Delta for the Manual Power Window Component with Central Locking System Feature

Automatic Power Window We define one delta applied to the core if the features *Automatic Power Window* and *Central Locking system* are selected for a product configuration. The delta *DAddAutoPWCLS* depicted in Fig. 4.23 transforms the corresponding core such that the window movement of the automatic power window is blocked by an active central locking system and unblocked based on the deactivation of the central locking system. Therefore, we add eight states and 16 transitions to integrate the new behavior into the core state machine test model.

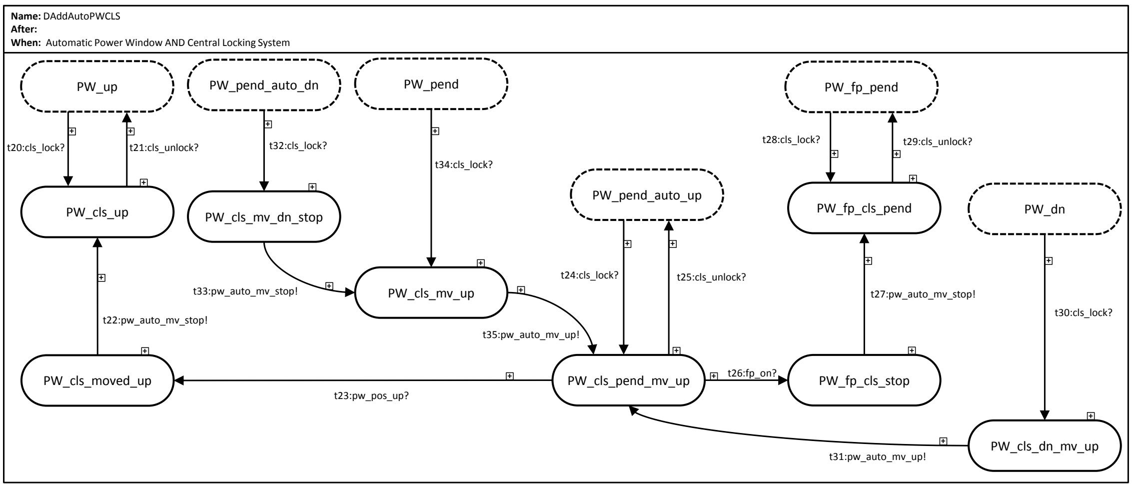

Figure 4.23.: State Machine Delta for the Automatic Power Window Component with Central Locking System Feature

Remote Control Key Controller We define one delta applied to the core if the feature *Safety Function* is selected for a product configuration. The delta *DAddRCKSF* shown in Fig. 4.24 transforms the corresponding core such that the remote control key controller re-locks the car after a timeout occurred representing the situation that the car was unintentionally unlocked. Therefore, we add four states and seven transitions, and remove one state and two transitions to integrate the new behavior into the core state machine test model.

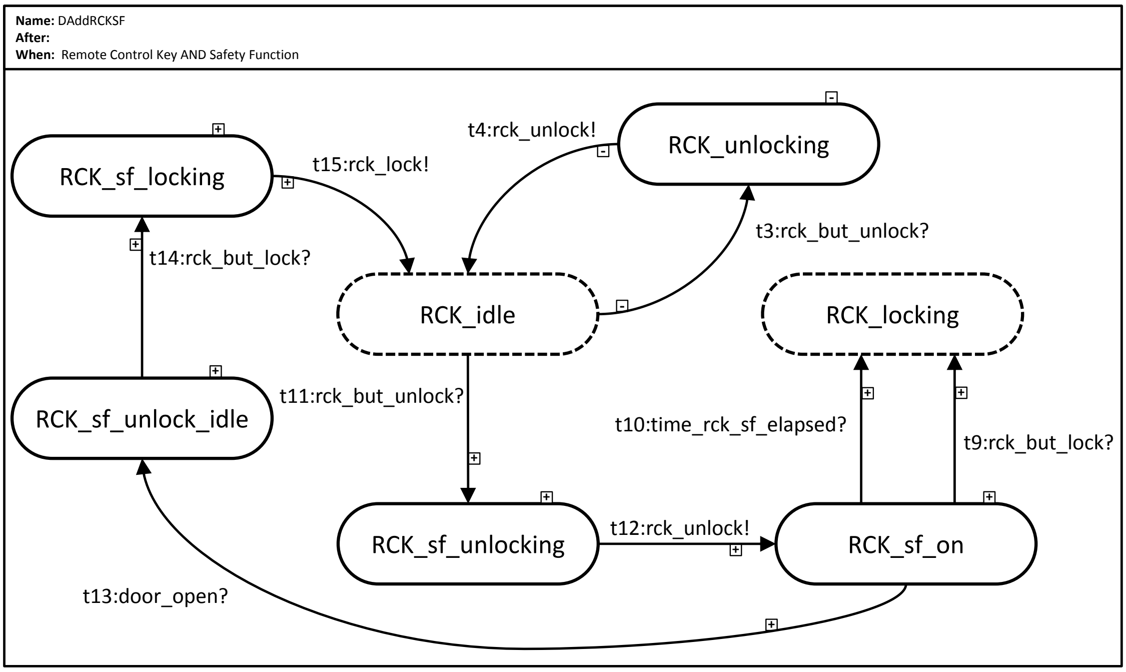

Figure 4.24.: State Machine Delta for the Remote Control Key Controller Component with Safety Function Feature

We define another delta applied to the core if the feature *Control Automatic Power Window* is selected for a product configuration. The delta *DAddRCKCAP* shown in Fig. 4.25 transforms the corresponding core such that the remote control key controller controls the upwards/-downwards movement of the window via the remote key. Therefore, we add two states and four transitions to integrate the new behavior into the core state machine test model.

We define another delta applied to the core if the features *Control Automatic Power Window* and *Safety Function* are selected for a product configuration. The delta *DAddRCKCAPSF* depicted in Fig. 4.26 further requires the application of the deltas *DAddRCKCAP* and *DAddRCKSF*. *DAddRCKCAPSF* transforms the corresponding modified core such that the remote control key controller controls the upwards/downwards movement of the window via the remote key in addition to the safety function. Therefore, we add four states and eight transitions to integrate the new behavior into the core state machine test model.

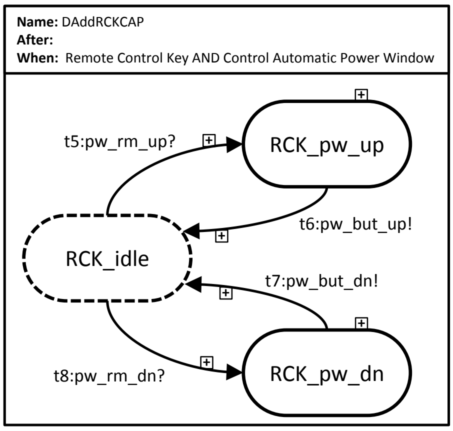

Figure 4.25.: State Machine Delta for the Remote Control Key Controller Component with Control Automatic Power Window Feature

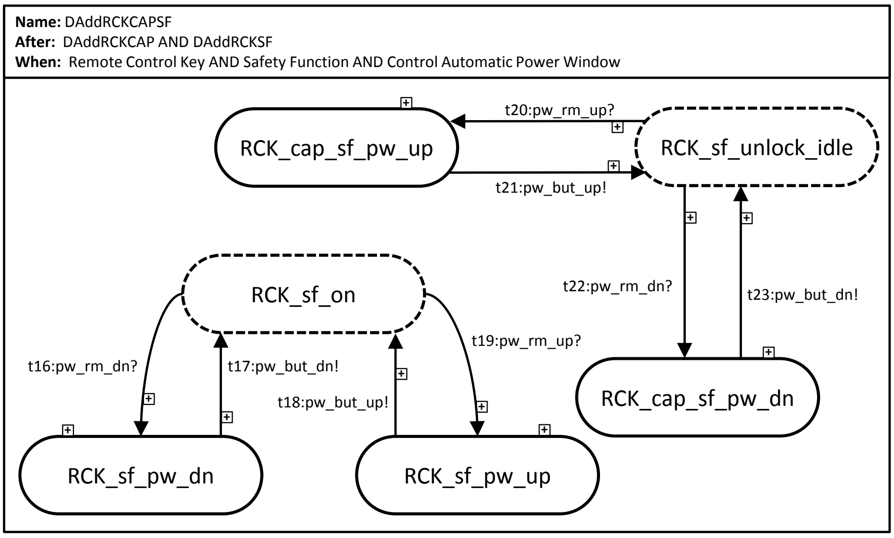

Figure 4.26.: State Machine Delta for the Remote Control Key Controller Component with Control Automatic Power Window and Safety Function Features

Alarm System We define one delta applied to the core if the feature *Control Alarm System* is selected for a product configuration. The delta *DAddASCAS* shown in Fig. 4.27 transforms the corresponding core such that the alarm monitoring of the alarm system is additionally enabled/disabled by a remote key. Therefore, we add three transitions to integrate the new behavior into the core state machine test model.

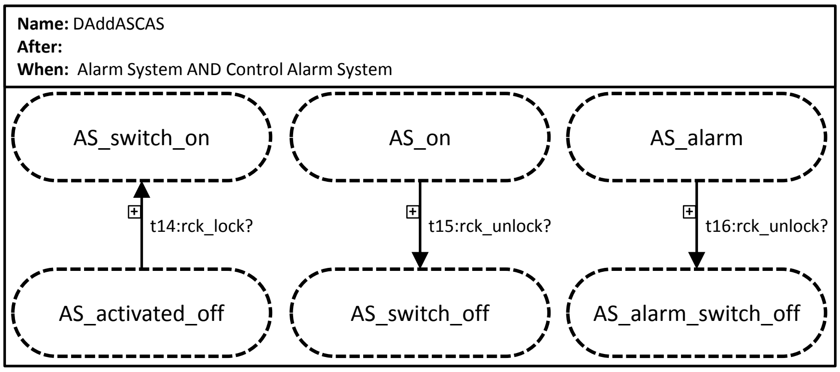

Figure 4.27.: State Machine Delta for the Alarm System Component with Control Alarm System Feature

We define another delta applied to the core if the feature *Interior Monitoring* is selected for a product configuration. The delta *DAddASIM* depicted in Fig. 4.28 transforms the corresponding core such that the alarm of the alarm system is triggered by the interior monitoring. Therefore, we add three states and six transitions, and remove two transitions to integrate the new behavior into the core state machine test model.

Exterior Mirror We define one delta applied to the core if the feature *Heatable* is selected for a product configuration. The delta *DAddEMHeating* shown in Fig. 4.29 transforms the corresponding core such that the exterior mirror is heatable if the outside temperature is too low. Therefore, we add 18 states and 36 transitions to integrate the new behavior into the core state machine test model.

We define another delta applied to the core if the feature *LED Exterior Mirror* is selected for a product configuration. The delta *DAddEMLEDEM* depicted in Fig. 4.30 and Fig. 4.31 transforms the corresponding core such that the exterior mirror sends the information of its current position to the corresponding LEDs. Therefore, we add 24 states and 48 transitions, and remove 24 transitions to integrate the new behavior into the core state machine test model.

Central Locking System We define one delta applied to the core if the feature *Automatic Locking* is selected for a product configuration. The delta *DAddCLSAL* shown in Fig. 4.32 transforms the corresponding core such that the central locking system locks the doors without blocking the window when the car is driving. Therefore, we add three states and four transitions to integrate the new behavior into the core state machine test model.

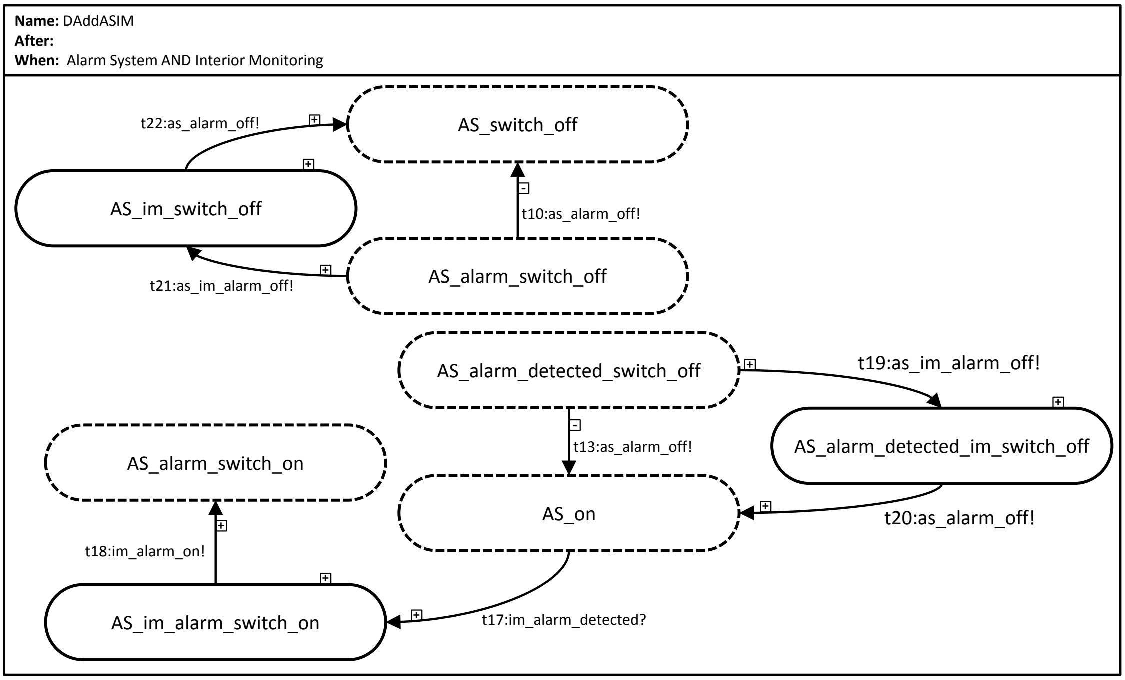

Figure 4.28.: State Machine Delta for the Alarm System Component with Interior Monitoring Feature

We define another delta applied to the core if the feature *Remote Control Key* is selected for a product configuration. The delta *DAddCLSRCK* depicted in Fig. 4.33 transforms the corresponding core such that the central locking system gets activated/deactivated via a remote key. Therefore, we add two transitions to integrate the new behavior into the core state machine test model.

Human Machine Interface We define one delta applied to the core if the feature *Alarm System* is selected for a product configuration. The delta *DAddHMIAS* shown in Fig. 4.34 transforms the corresponding core such that the human machine interface enables the activation/deactivation of the alarm system via the interaction with the driver. Therefore, we add two states and four transitions to integrate the new behavior into the core state machine test model.

We define another delta applied to the core if the features *Alarm System* and *LED Alarm System* are selected for a product configuration. The delta *DAddHMILEDAS* depicted in Fig. 4.35 transforms the corresponding core such that the human machine interface enables the confirmation of the silent alarm. Therefore, we add one state and two transitions to integrate the new behavior into the core state machine test model.

We define another delta applied to the core if the features *Manual Power Window* and *LED Power Window* are selected for a product configuration. The delta *DAddHMILEDManPW* shown in Fig. 4.36 transforms the corresponding core such that the human machine interface provides information about the release of the window buttons for the corresponding

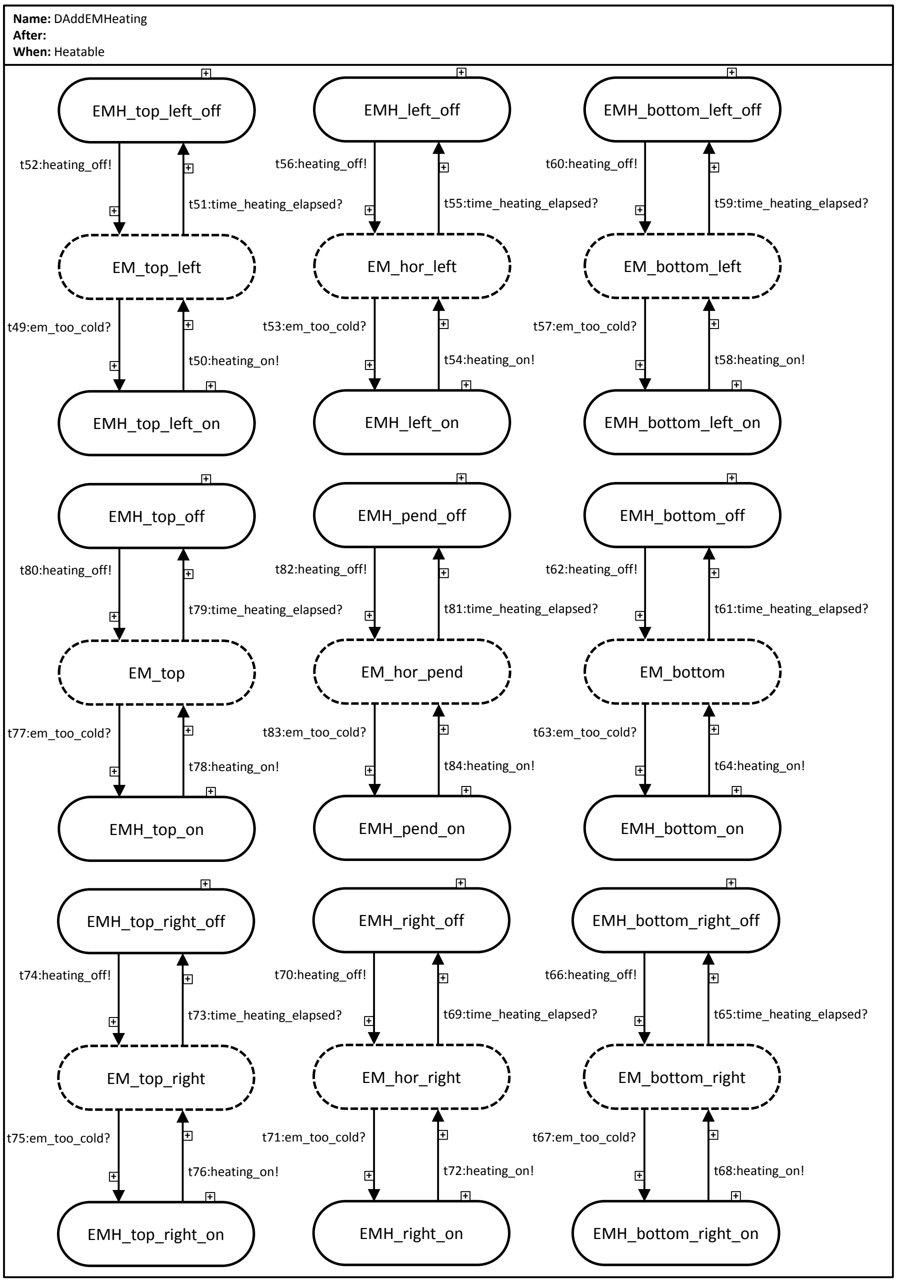

Figure 4.29.: State Machine Delta for the Exterior Mirror Component with Heatable Feature

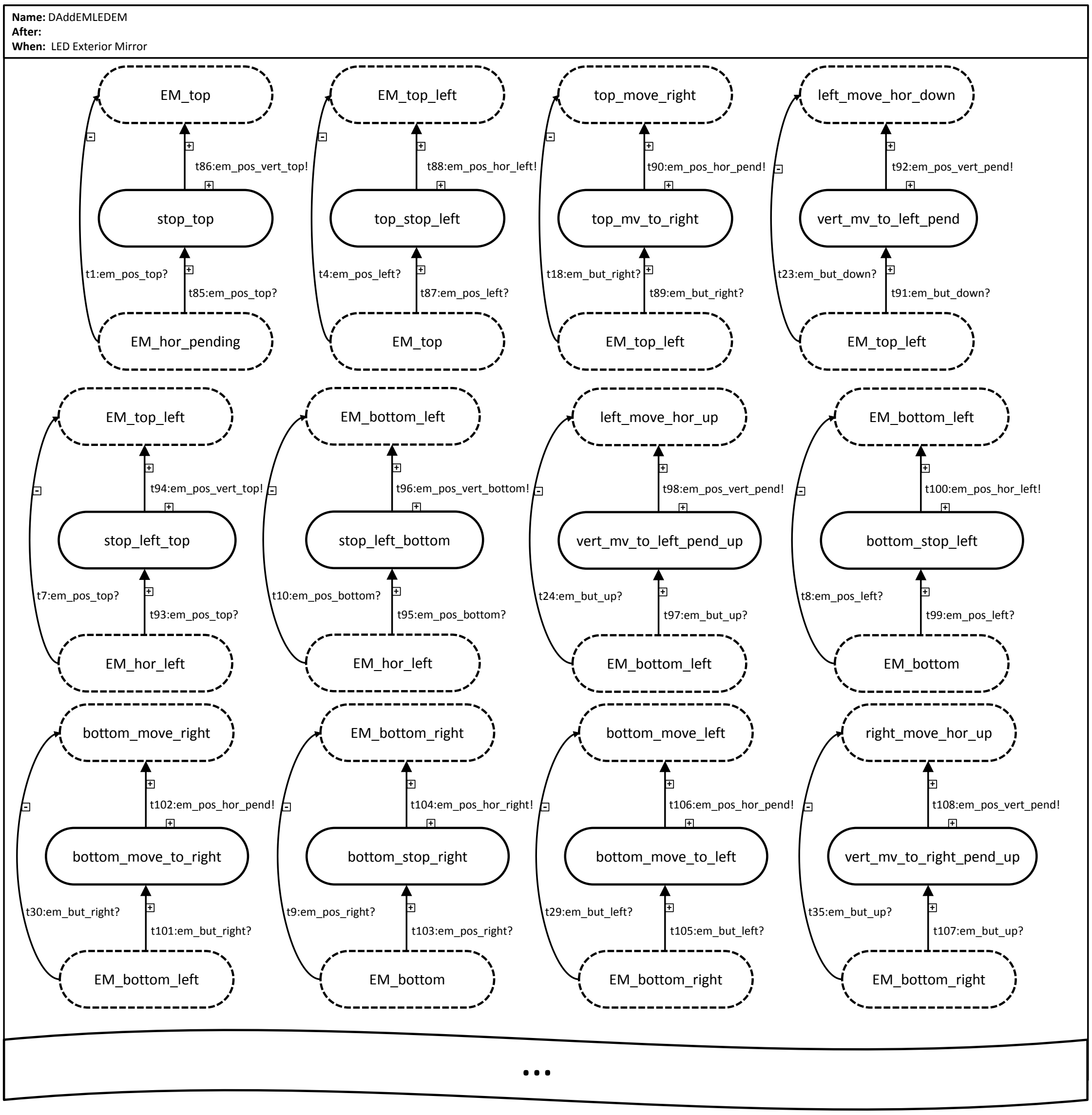

Figure 4.30.: State Machine Delta for the Exterior Mirror Component with LED Exterior Mirror Feature (1)

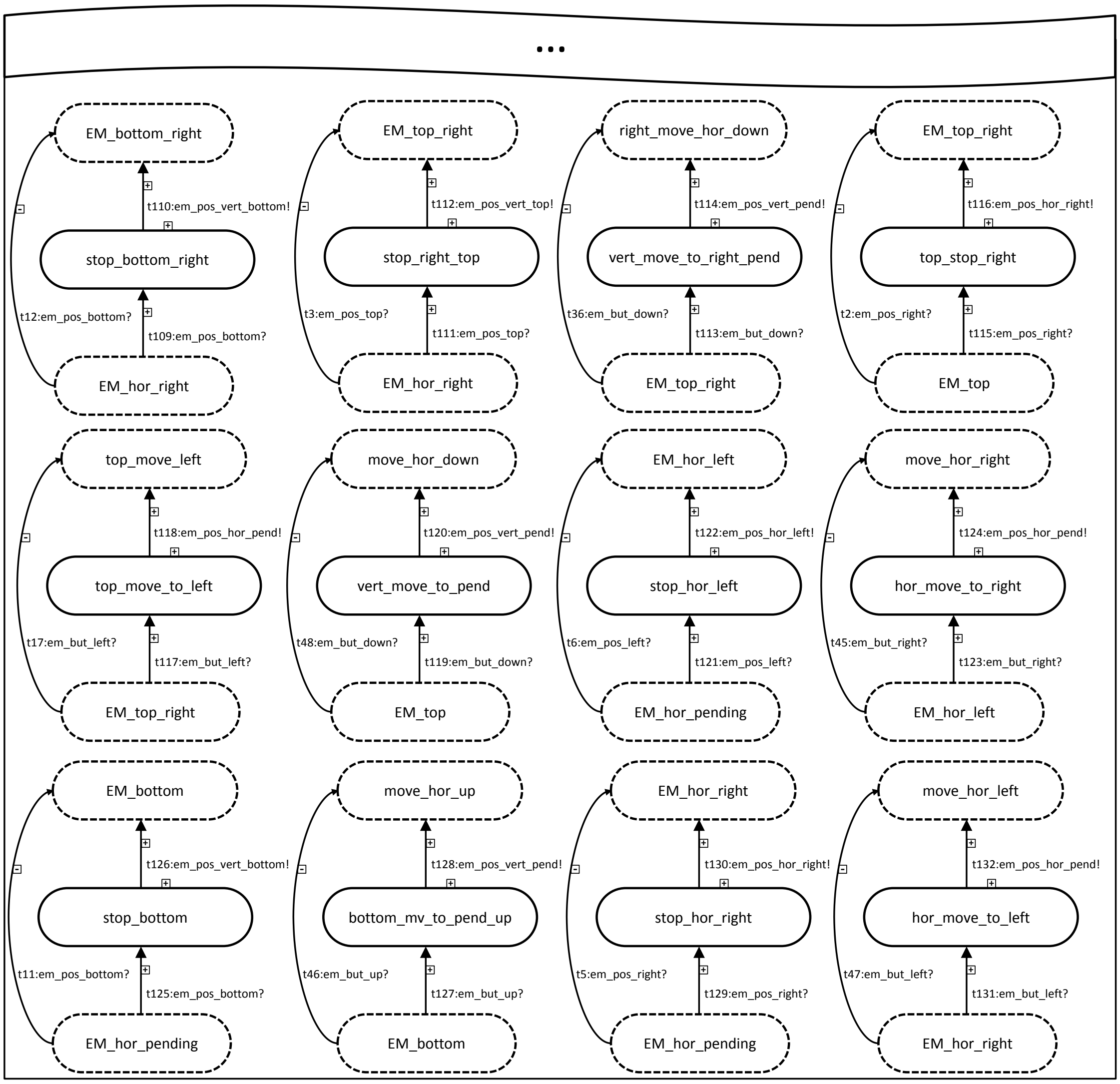

Figure 4.31.: State Machine Delta for the Exterior Mirror Component with LED Exterior Mirror Feature (2)

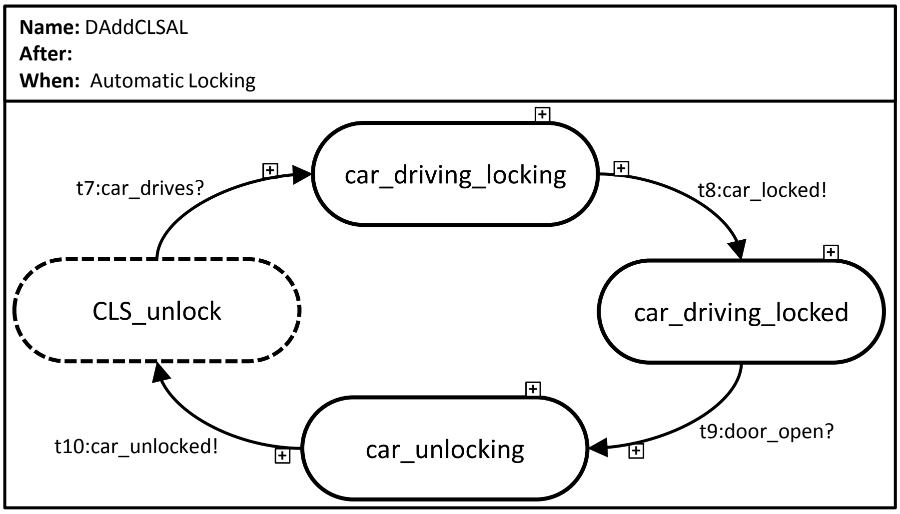

Figure 4.32.: State Machine Delta for the Central Locking System Component with Automatic Locking Feature

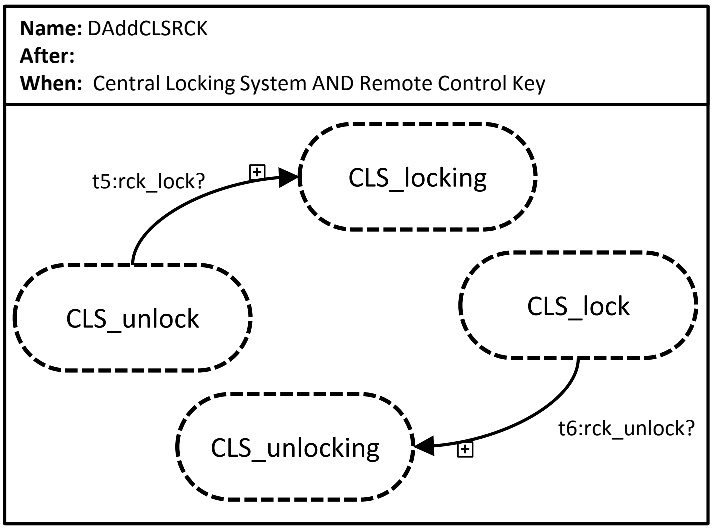

Figure 4.33.: State Machine Delta for the Central Locking System Component with Remote Control Key Feature

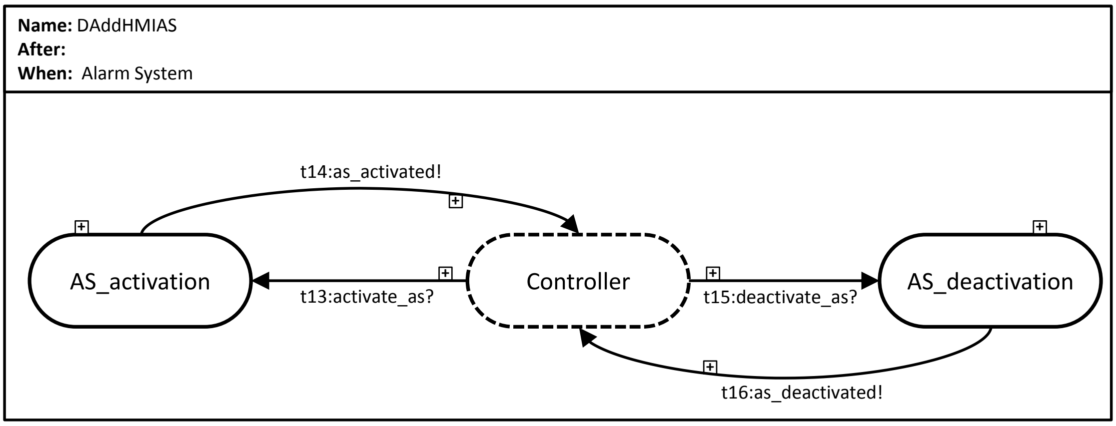

Figure 4.34.: State Machine Delta for the Human Machine Interface Component with Alarm System Feature

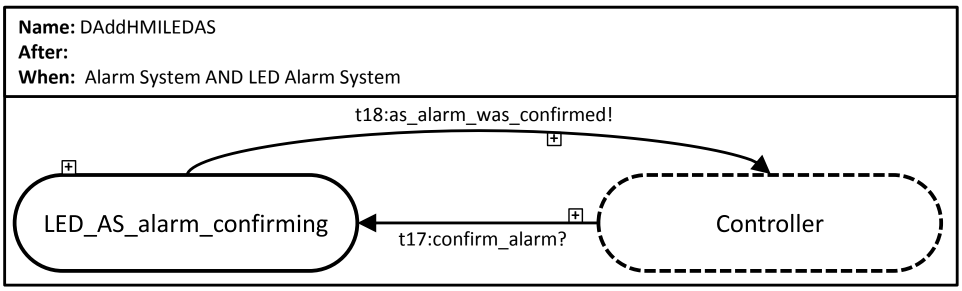

Figure 4.35.: State Machine Delta for the Human Machine Interface Component with LED Alarm System Feature

LEDs. Therefore, we add one state and three transitions to integrate the new behavior into the core state machine test model.

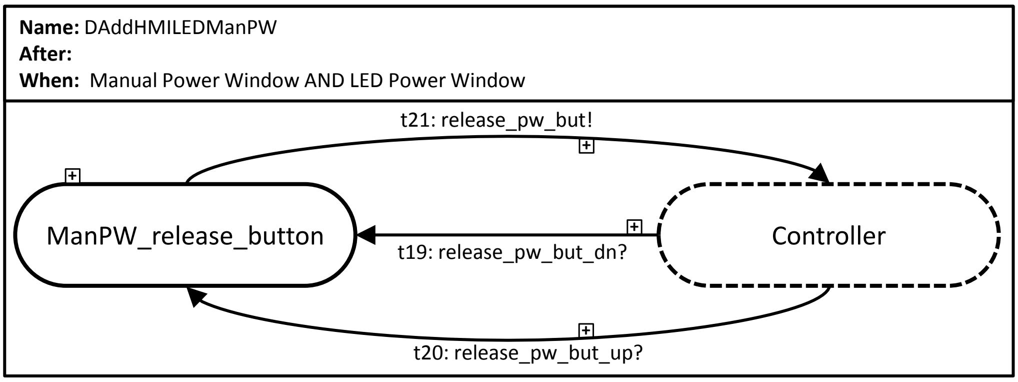

Figure 4.36.: State Machine Delta for the Human Machine Interface Component with Manual Power Window and LED Power Window Features

LED Automatic Power Window We define one delta applied to the core if the features *Automatic Power Window*, *LED Power Window* and *Central Locking System* are selected for a product configuration. The delta *DAddLEDAutoPWCLS* depicted in Fig. 4.37 transform the corresponding core such that there is a new LED turning on/off if the automatic power window moves up while the central locking system is active. Therefore, we add six states and nine transitions to integrate the new behavior into the core state machine test model.

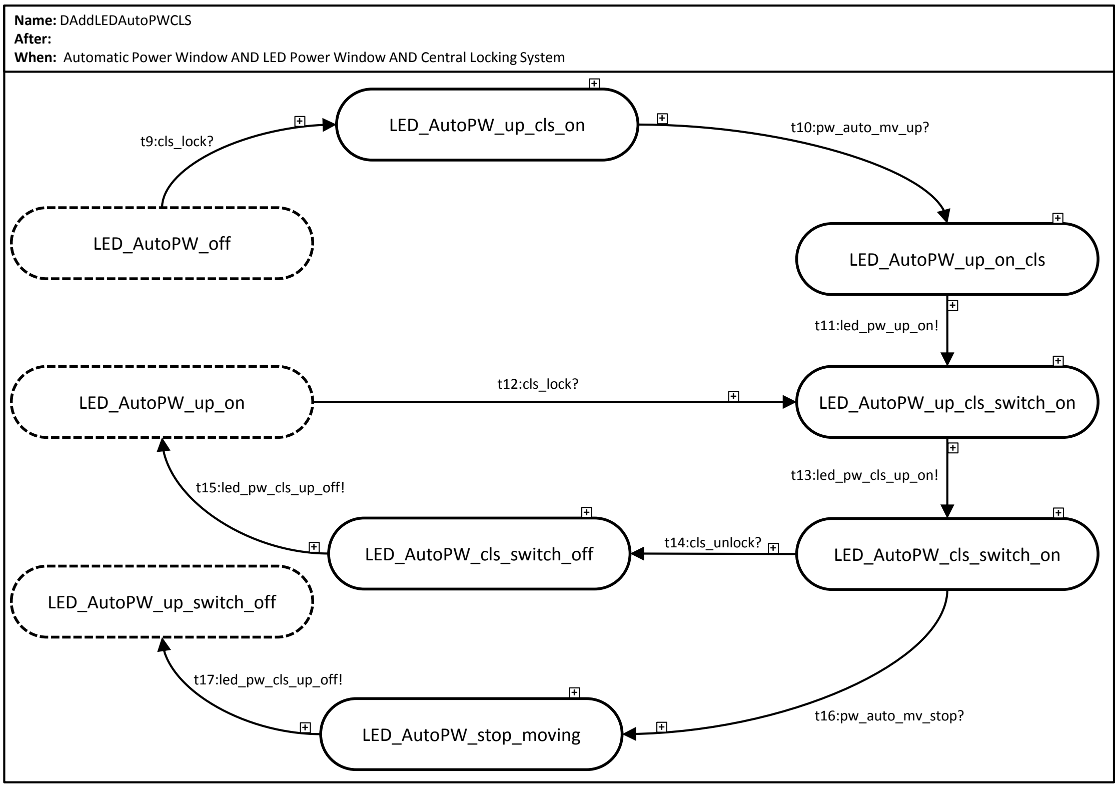

Figure 4.37.: State Machine Delta for the LED Automatic Power Window Component with Central Locking System Feature

4.3. State Machine Test Model Variants

In this section, the state machine test models for the various component variants described in Sect. 3.1 are provided. Based on the definition of the core test models and the state machine delta documented above, we modeled 20 further variants as follows.

Manual Power Window with Central Locking System The state machine test model variant for the component *Manual Power Window* with *Central Locking System* is shown in Fig. 4.38, comprising 13 states and 27 transitions. The test model variant is obtained by applying the delta *DAddManPWCLS* to the corresponding core. In addition to the core functionality described above, the test model variant specifies further the blocking of the downwards movement of the window if the central locking system is active (*cls_lock*). In this case, the window is still able to move upwards. The downwards movement is unblocked if the central locking system gets inactive (*cls_unlock*), respectively.

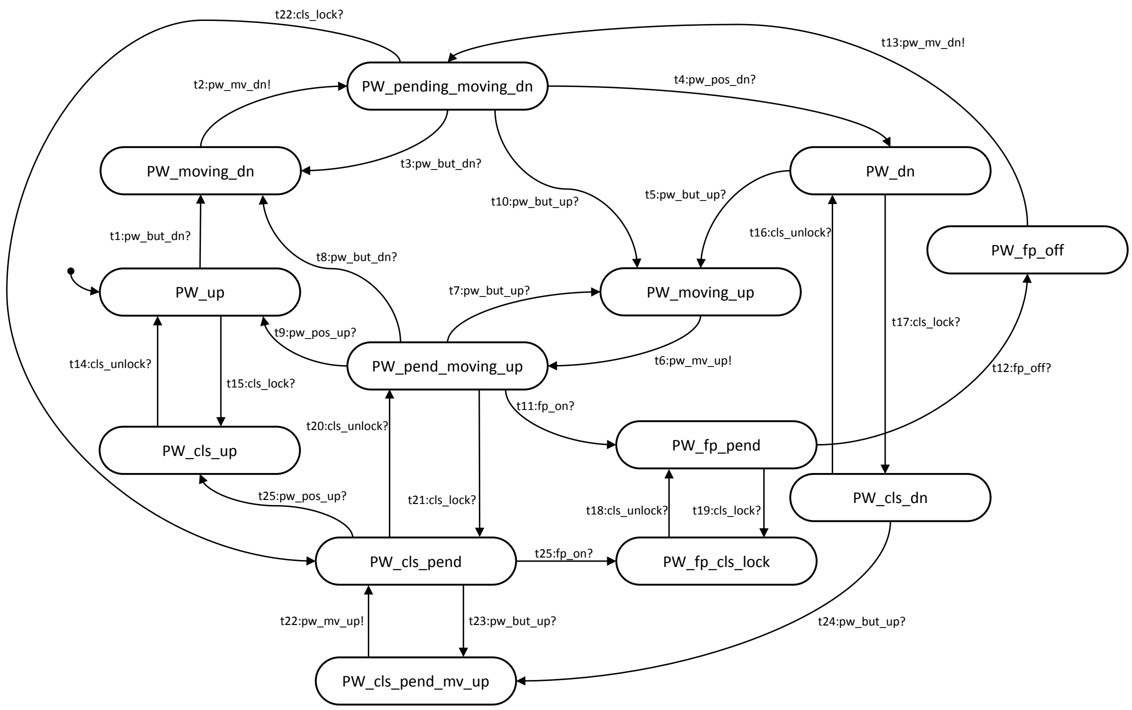

Figure 4.38.: Behavioral Specification of the Manual Power Window Component Variant with Central Locking System Feature

Automatic Power Window with Central Locking System The state machine test model variant for the component *Automatic Power Window* with *Central Locking System* is depicted in Fig. 4.39, comprising 23 states and 35 transitions. The test model variant is obtained by applying the delta *DAddAutoPWCLS* to the corresponding core. In addition to the core functionality described above, the test model variant specifies further the blocking of the automated downwards movement of the window when the central locking system is active (*cls_lock*). In this case, the window is automatically moved upwards. The downwards movement is unblocked if the central locking system gets inactive (*cls_unlock*), respectively.

Exterior Mirror with Heatable The state machine test model variant for the component *Exterior Mirror* with *Heatable* is shown in Fig. 4.40, comprising 39 states and 84 transitions. The test model variant is obtained by applying the delta *DAddEMHeating* to the corresponding core. In addition to the core functionality described above, the test model variant specifies further the activation (*heating_on*) of the mirror heater if the outside temperature is too low (*em_too_cold*). The heater activation is possible in each position, i.e., at all times. The heater activation is controlled by a timeout such that it is deactivated (*heating_off*) if the heating time elapsed (*time_heating_elapsed*).

Exterior Mirror with LED Exterior Mirror The state machine test model variant for the component *Exterior Mirror* with *LED Exterior Mirror* is depicted in Fig. 4.41, comprising 45 states and 72 transitions. The test model variant is obtained by applying the delta *DAddEMLEDEM* to the corresponding core. In addition to the core functionality described above, the test model variant specifies further the provision of the current position for the corresponding LEDs by sending the information about the position in the vertical (*em_pos_vert_top*, *em_pos_vert_pend*, *em_pos_vert_bottom*) as well as in the horizontal (*em_pos_hor_left*, *em_pos_hor_pend*, *em_pos_hor_right*) axes.

Exterior Mirror with Heatable and LED Exterior Mirror The state machine test model variant for the component *Exterior Mirror* with *Heatable* and *LED Exterior Mirror* is shown in Fig. 4.42, comprising 63 states and 108 transitions. The test model variant is obtained by applying the deltas *DAddEMHeating* and *DAddEMLEDEM* to the corresponding core. In addition to the core functionality described above, the test model variant specifies further the provision of the current position for the corresponding LEDs and the activation/deactivation of the mirror heater.

Alarm System with Control Alarm System The state machine test model variant for the component *Alarm System* with *Control Alarm System* is depicted in Fig. 4.43, comprising 10 states and 17 transitions. The test model variant is obtained by applying the delta *DAddASCAS* to the corresponding core. In addition to the core functionality described above, the test model variant specifies further the enabling/disabling of the alarm monitoring of the alarm system controlled by the remote key (*rck_lock*, *rck_unlock*).

Alarm System with Interior Monitoring The state machine test model variant for the component *Alarm System* with *Interior Monitoring* is shown in Fig. 4.44, comprising 13 states and 17 transitions. The test model variant is obtained by applying the delta *DAddASIM* to the corresponding core. In addition to the core functionality described above, the test model variant specifies further the triggering of the interior alarm (*im_alarm_on*) if a unknown motion or object is detected inside the car (*im_alarm_detected*). By either unlocking the car or elapsing the alarm time, the interior alarm is stopped (*im_alarm_off*).

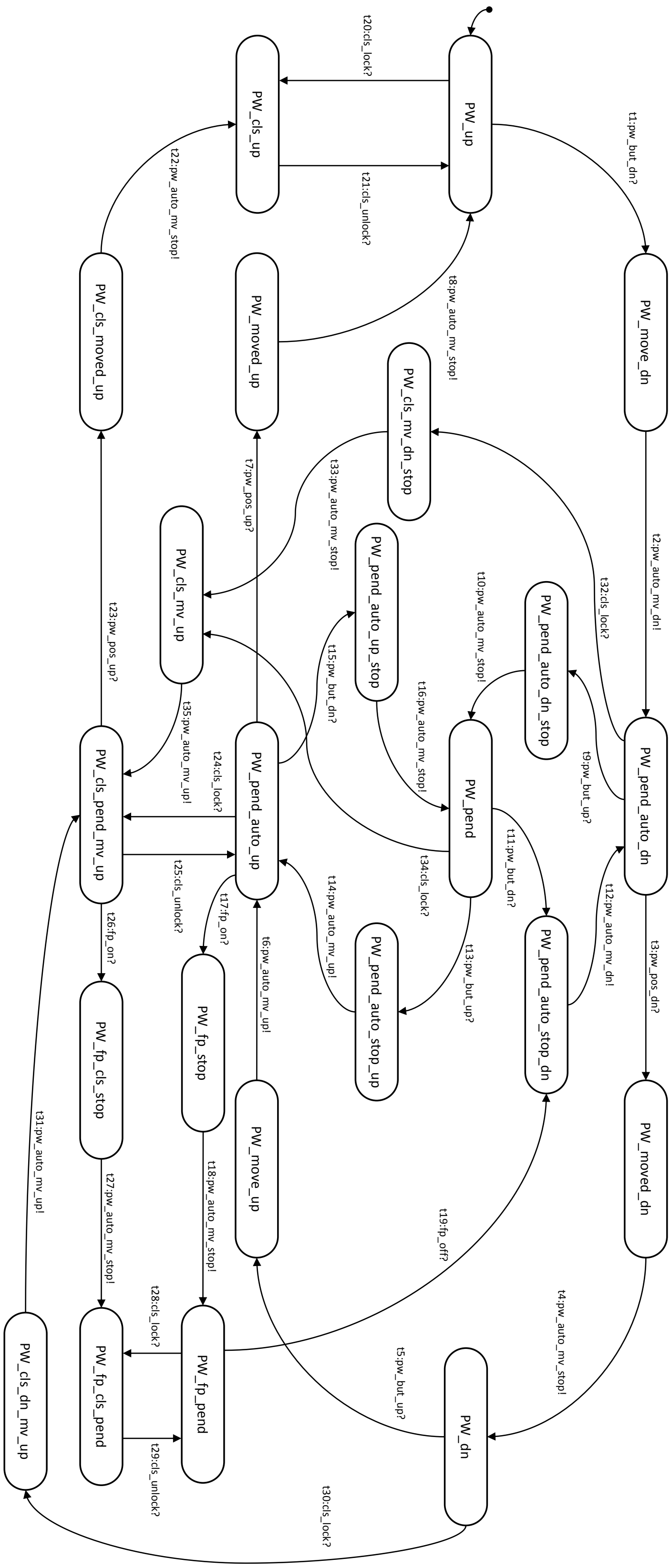

Figure 4.39.: Behavioral Specification of the Automatic Power Window Component Variant with Central Locking System Feature

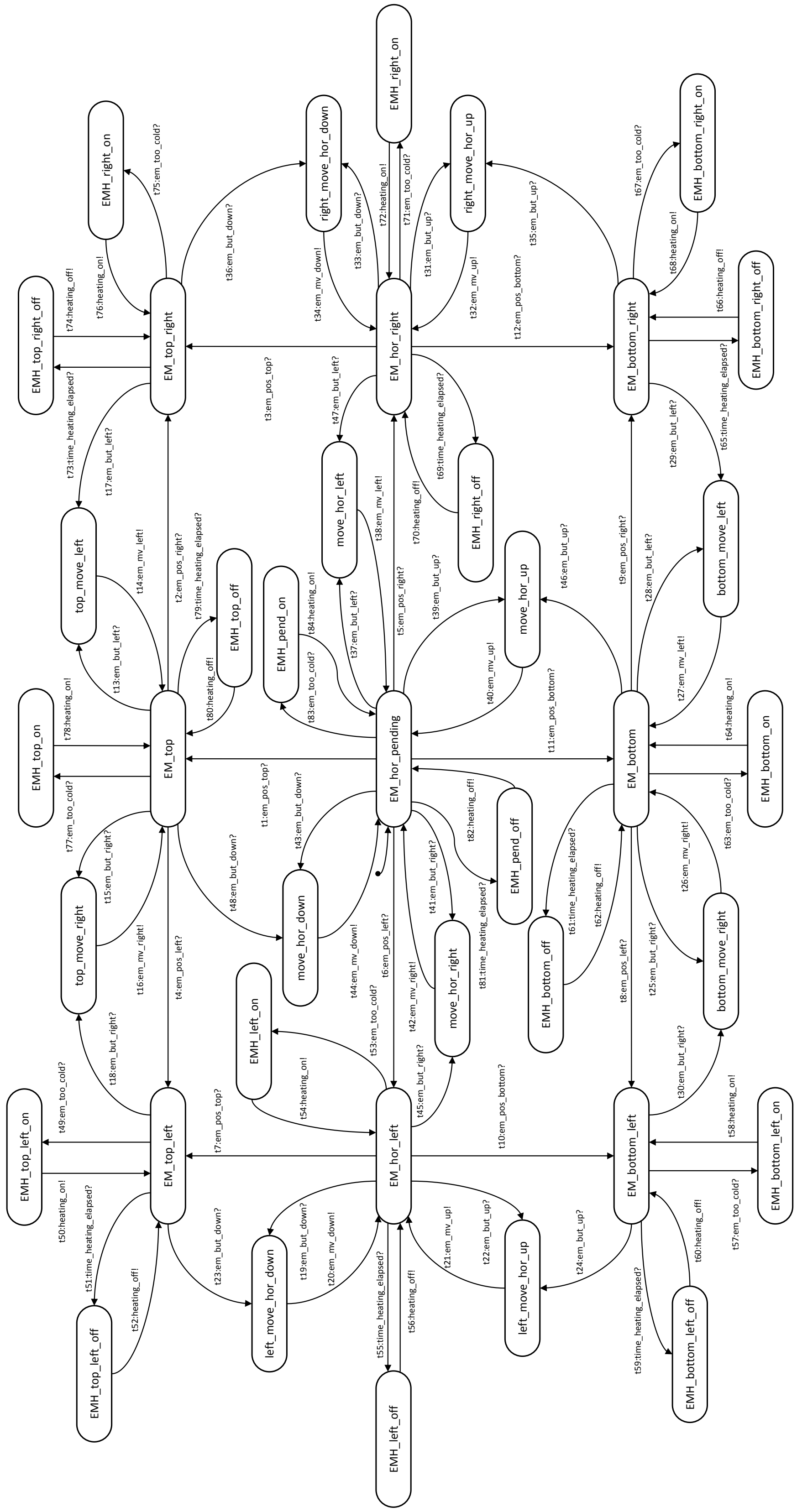

Figure 4.40.: Behavioral Specification of the Exterior Mirror Component Variant with Heatable Feature

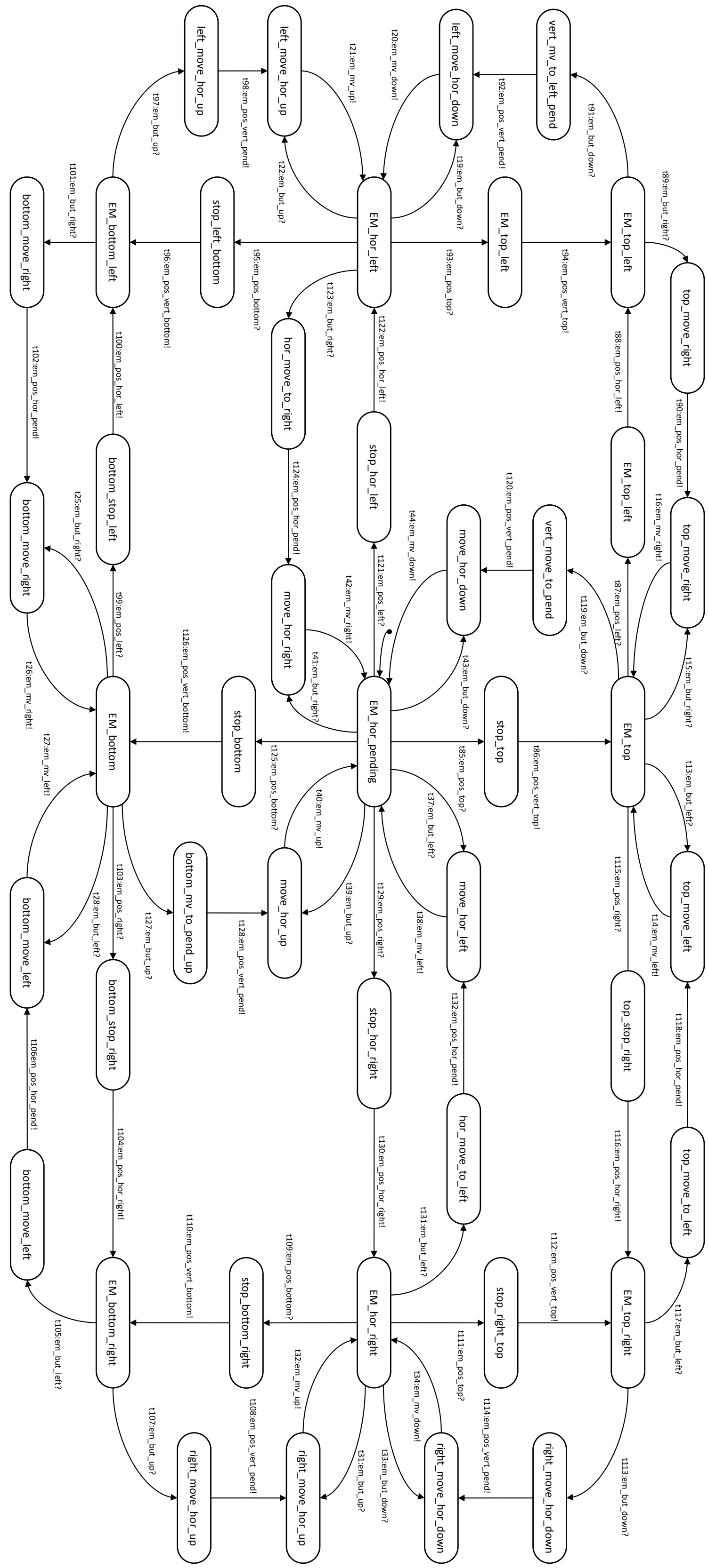

Figure 4.41.: Behavioral Specification of the Exterior Mirror Component Variant with LED Exterior Mirror Feature

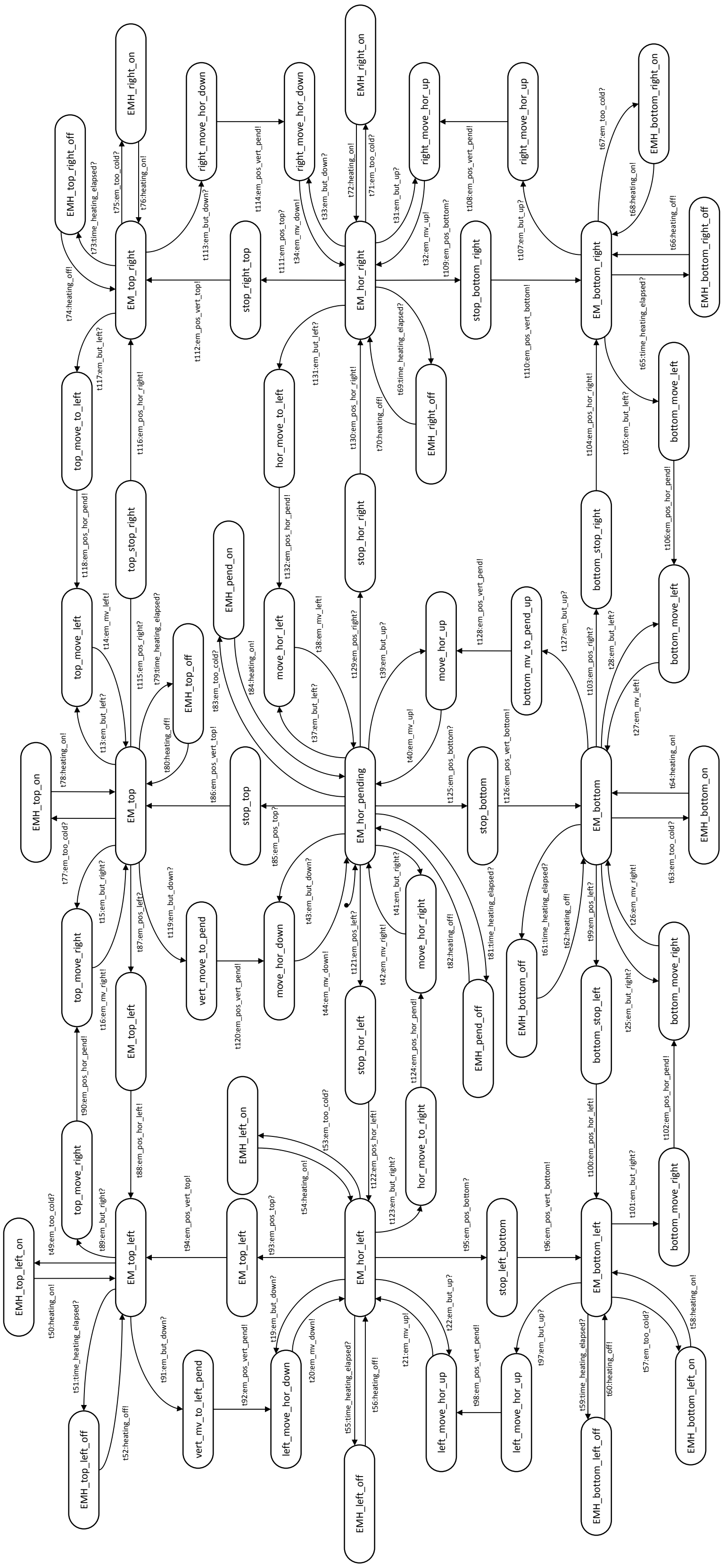

Figure 4.42.: Behavioral Specification of the Component Exterior Mirror Component Variant with LED Exterior Mirror and Heatable Features

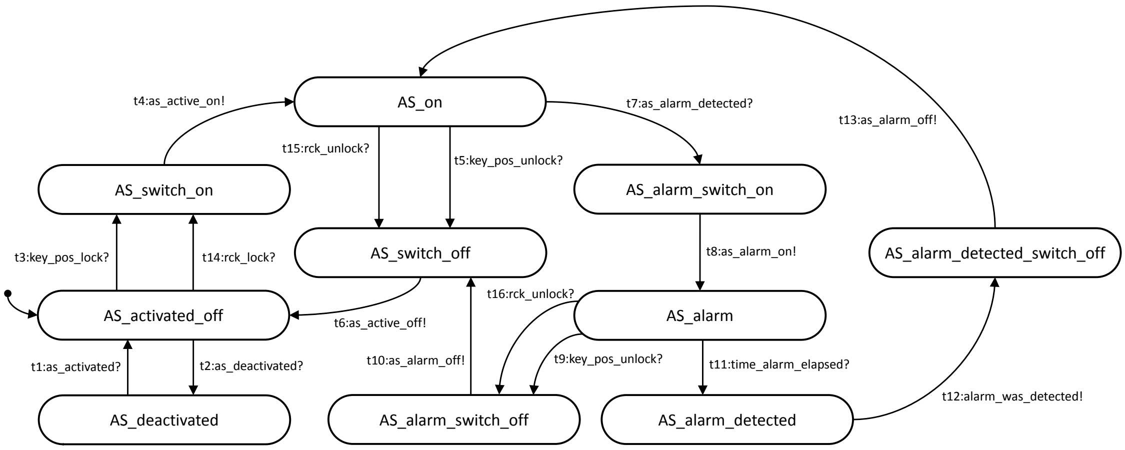

Figure 4.43.: Behavioral Specification of the Alarm System Component Variant with Control Alarm System Feature

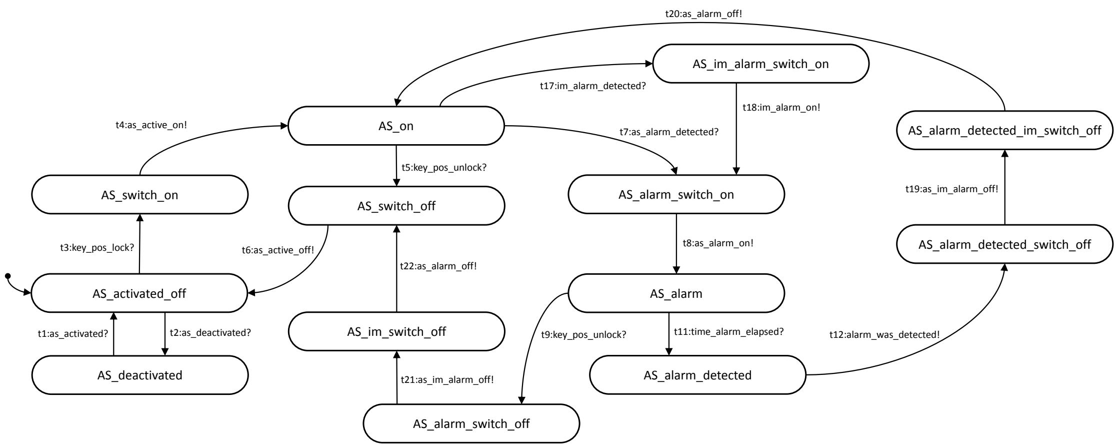

Figure 4.44.: Behavioral Specification of the Alarm System Component Variant with Interior Monitoring Feature

Alarm System with Control Alarm System and Interior Monitoring The state machine test model variant for the component *Alarm System* with *Control Alarm System* and *Interior Monitoring* is depicted in Fig. 4.45, comprising 13 states and 20 transitions. The test model variant is obtained by applying the deltas *DAddASCAS* and *DAddASIM* to the corresponding core. In addition to the core functionality described above, the test model variant specifies further the triggering of the interior alarm if something is detected inside the car as well as the enabling/disabling of the alarm monitoring of the alarm system controlled by the remote key.

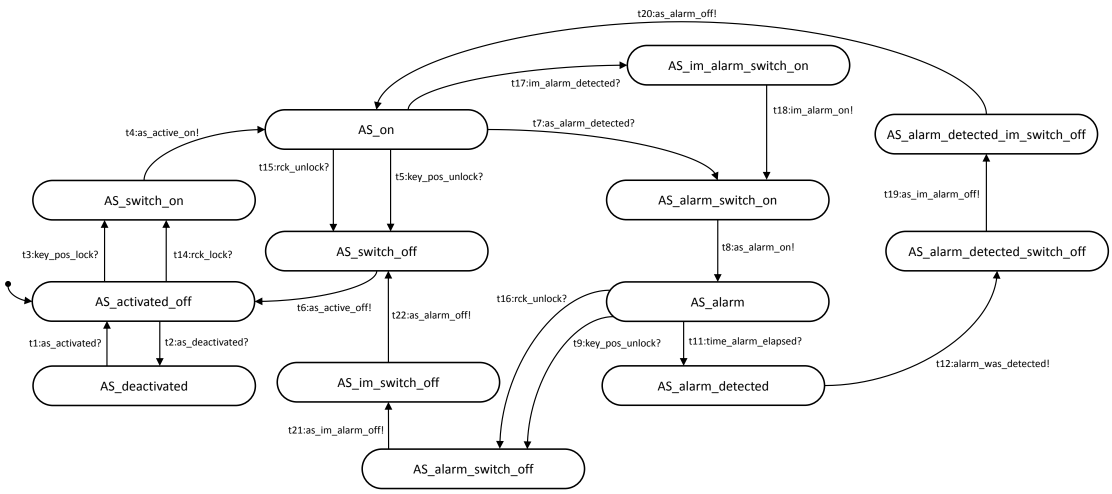

Figure 4.45.: Behavioral Specification of the Alarm System Component Variant with Control Alarm System and Interior Monitoring Features

Remote Control Key Controller with Control Automatic Power Window The state machine test model variant for the component *Remote Control Key Controller* with *Control Automatic Power Window* is shown in Fig. 4.46, comprising five states and eight transitions. The test model variant is obtained by applying the delta *DAddRCKCAP* to the corresponding core. In addition to the core functionality described above, the test model variant specifies further the control of the window upwards/downwards movement (*pw_but_up*, *pw_but_dn*) via the remote key (*pw_rm_up*, *pw_rm_up*).

Remote Control Key Controller with Safety Function The state machine test model variant for the component *Remote Control Key Controller* with *Safety Function* is depicted in Fig. 4.47, comprising six states and nine transitions. The test model variant is obtained by applying the delta *DAddRCKSF* to the corresponding core. In addition to the core functionality described above, the test model variant specifies further the automatic locking of the car after a timeout occurred (*time_sf_elapsed*) representing the scenario that the car was unintentionally unlocked. By opening the door (*door_open*), the timeout is stopped and the car remains unlocked.

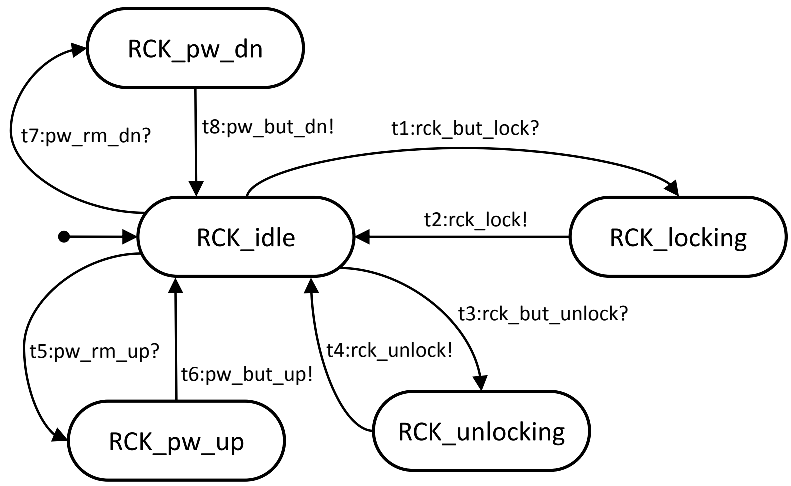

Figure 4.46.: Behavioral Specification of the Remote Control Key Controller Component Variant with Control Automatic Power Window Feature

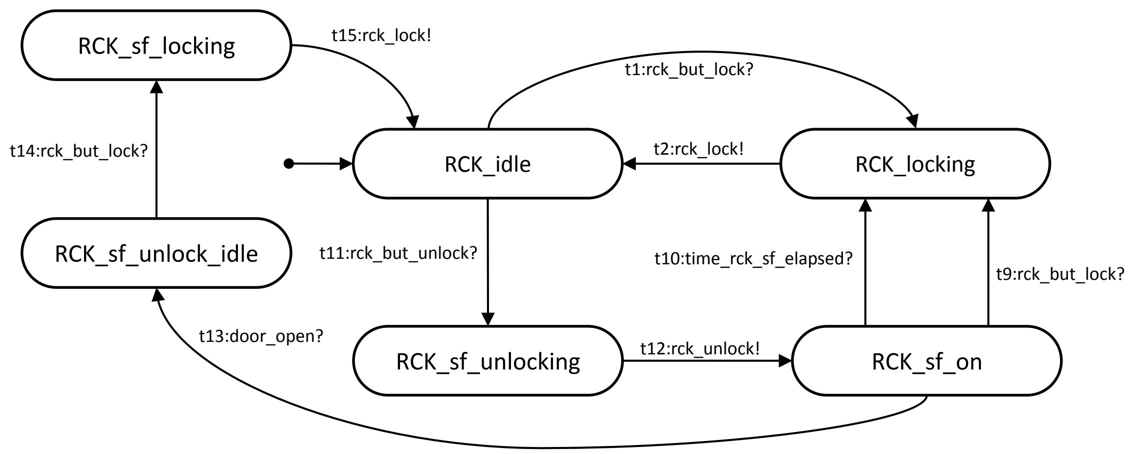

Figure 4.47.: Behavioral Specification of the Remote Control Key Controller Component Variant with Safety Function Feature

Remote Control Key Controller with Control Automatic Power Window and Safety Function The state machine test model variant for the component *Remote Control Key Controller* with *Control Automatic Power Window* and *Safety Function* is shown in Fig. 4.48, comprising 12 states and 21 transitions. The test model variant is obtained by applying the deltas *DAddRCKCAP*, *DAddRCKSF* and *DAddRCKCAPSF* to the corresponding core. In addition to the core functionality described above, the test model variant specifies further the control of the window upwards/downwards movement via the remote key as well as the automatic locking of the car after a timeout occurred (*time_sf_elapsed*) representing the scenario that the car was unintentionally unlocked.

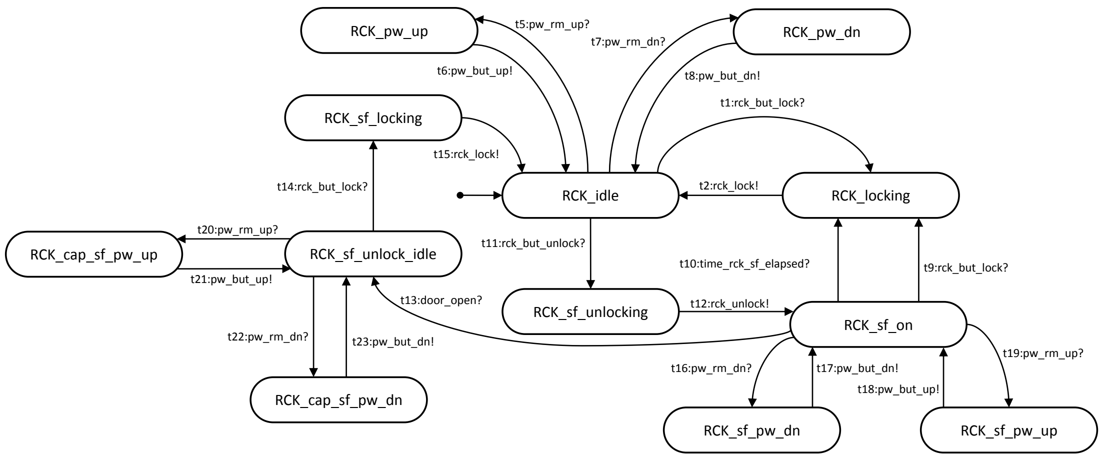

Figure 4.48.: Behavioral Specification of the Remote Control Key Controller Component Variant with Control Automatic Power Window and Safety Function Features

Central Locking System with Automatic Locking The state machine test model variant for the component *Central Locking System* with *Automatic Locking* is depicted in Fig. 4.49, comprising seven states and eight transitions. The test model variant is obtained by applying the delta *DAddCLSAL* to the corresponding core. In addition to the core functionality described above, the test model variant specifies further the automatic locking of the car (*car_locked*) without blocking the window movement when the car is driving (*car_drives*). By opening the doors (*door_open*), the car gets unlocked (*car_unlocked*).

Central Locking System with Remote Control Key The state machine test model variant for the component *Central Locking System* with *Remote Control Key* is shown in Fig. 4.50, comprising four states and six transitions. The test model variant is obtained by applying the delta *DAddCLSRCK* to the corresponding core. In addition to the core functionality described above, the test model variant specifies further the locking/unlocking of the central locking system controlled by the remote key (*rck_lock*, *rck_unlock*).

Central Locking System with Remote Control Key and Automatic Locking The state machine test model variant for the component *Central Locking System* with *Remote Control Key* and *Automatic Locking* is depicted in Fig. 4.51, comprising seven states and 10 transitions. The

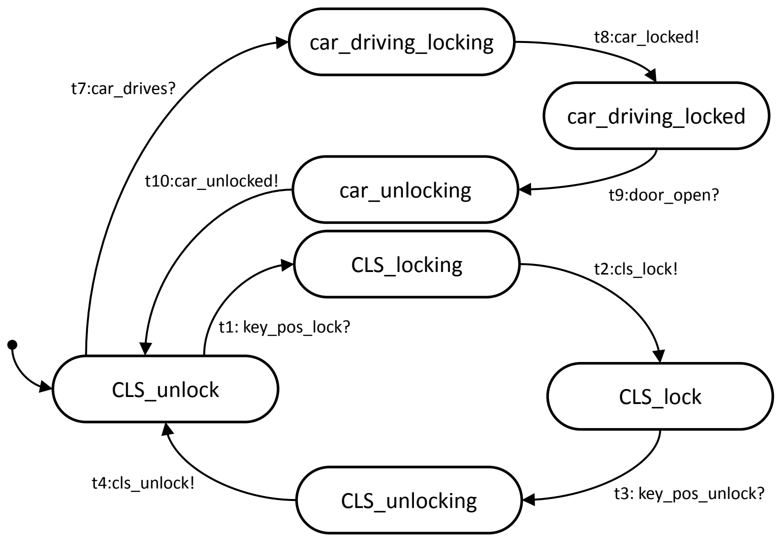

Figure 4.49.: Behavioral Specification of the Central Locking System Component Variant with Automatic Locking Feature

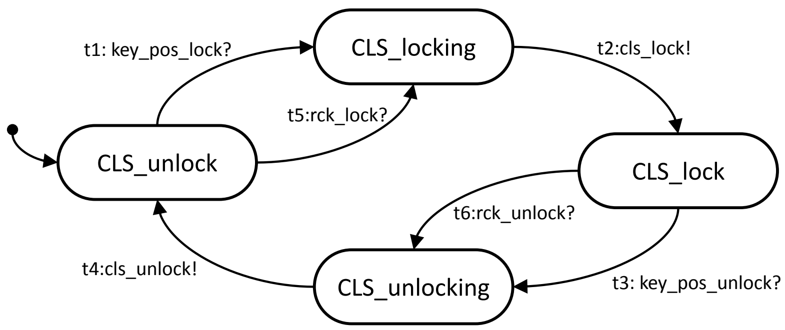

Figure 4.50.: Behavioral Specification of the Central Locking System Component Variant with Remote Control Key Feature

test model variant is obtained by applying the deltas *DAddCLSRCK* and *DAddCLSAL* to the corresponding core. In addition to the core functionality described above, the test model variant specifies further the locking/unlocking of the central locking system controlled by the remote key as well as the automated locking of the car when the car is driving.

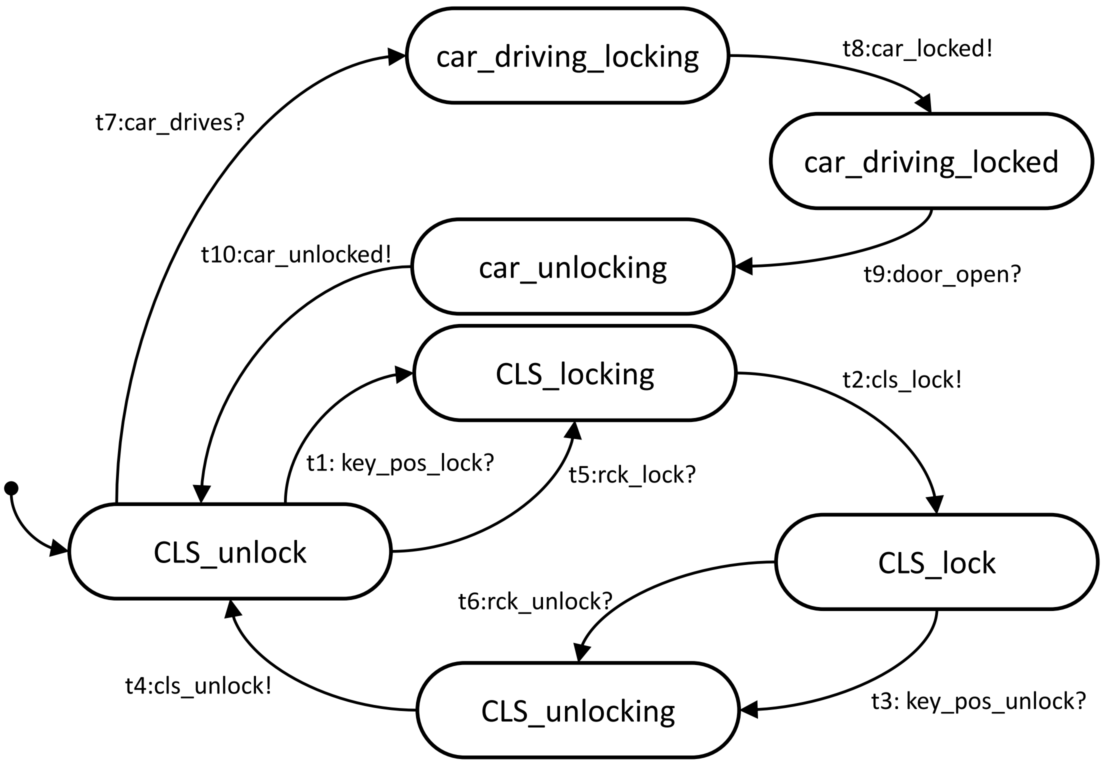

Figure 4.51.: Behavioral Specification of the Central Locking System Component Variant with Remote Control Key and Automatic Locking Features

Human Machine Interface with Manual Power Window and LED Power Window The state machine test model variant for the component *Human Machine Interface* with *Manual Power Window* and *LED Power Window* is shown in Fig. 4.52, comprising eight states and 15 transitions. The test model variant is obtained by applying the delta *DAddHMILEDManPW* to the corresponding core. In addition to the core functionality described above, the test model variant specifies further the provision of the release state of the window buttons (*release_pw_but_up*, *release_pw_but_dn*) for the corresponding LED by sending the release information (*release_pw_but*).

Human Machine Interface with Alarm System The state machine test model variant for the component *Human Machine Interface* with *Alarm System* is depicted in Fig. 4.53, comprising nine states and 16 transitions. The test model variant is obtained by applying the delta *DAddHMIAS* to the corresponding core. In addition to the core functionality described above, the test model variant specifies further the provision of the activation/deactivation of the

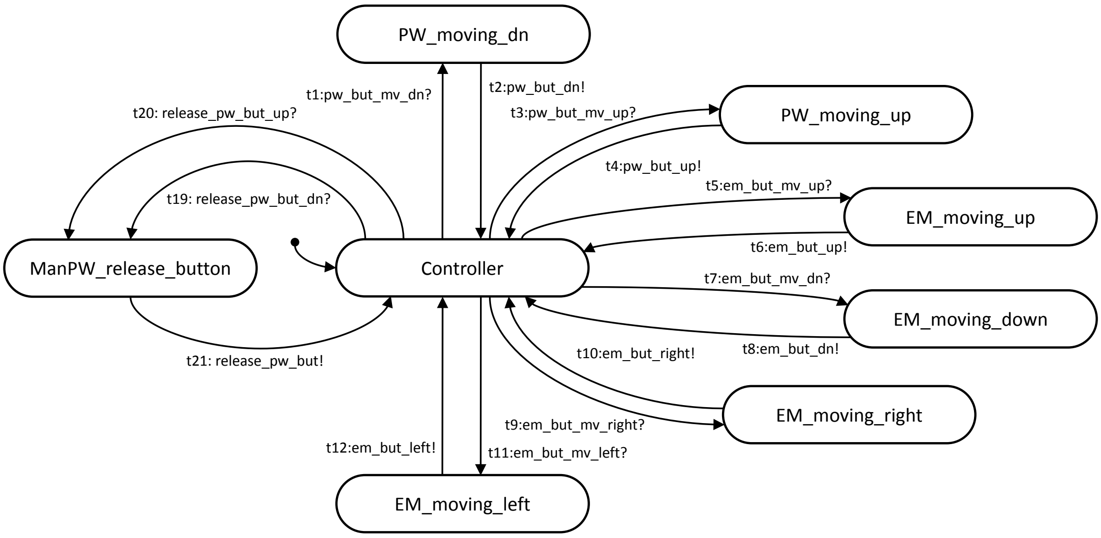

Figure 4.52.: Behavioral Specification of the Human Machine Interface Component Variant with Manual Power Window and LED Power Window Features

alarm system (*as_activated*, *as_deactivated*) controllable by the driver (*activate_as*, *deactivate_as*) via the human machine interface.

Human Machine Interface with Alarm System and LED Alarm System The state machine test model variant for the component *Human Machine Interface* with *Alarm System* and *LED Alarm System* is shown in Fig. 4.54, comprising 10 states and 18 transitions. The test model variant is obtained by applying the deltas *DAddHMIAS* and *DAddHMILEDAS* to the corresponding core. In addition to the core functionality described above, the test model variant specifies further the provision of the activation/deactivation of the alarm system as well as the confirmation (*as_alarm_was_confirmed*) of the silent alarm based on the interaction with the driver (*confirm_alarm*).

Human Machine Interface with Alarm System, Manual Power Window and LED Power Window The state machine test model variant for the component *Human Machine Interface* with *Alarm System*, *Manual Power Window* and *LED Power Window* is depicted in Fig. 4.55, comprising 10 states and 19 transitions. The test model variant is obtained by applying the deltas *DAddHMIAS* and *DAddHMILEDManPW* to the corresponding core. In addition to the core functionality described above, the test model variant specifies further the provision of the activation/deactivation of the alarm system as well as the provision of the release state of the window buttons for the corresponding LED.

Human Machine Interface with Alarm System, LED Alarm System, Manual Power Window and LED Power Window The state machine test model variant for the component *Human Machine Interface* with *Alarm System*, *LED Alarm System Manual Power Window* and *LED Power Window* is shown in Fig. 4.56, comprising 11 states and 21 transitions. The test model variant is obtained by applying the deltas *DAddHMIAS*, *DAddHMILEDAS* and *DAddHMILED-*

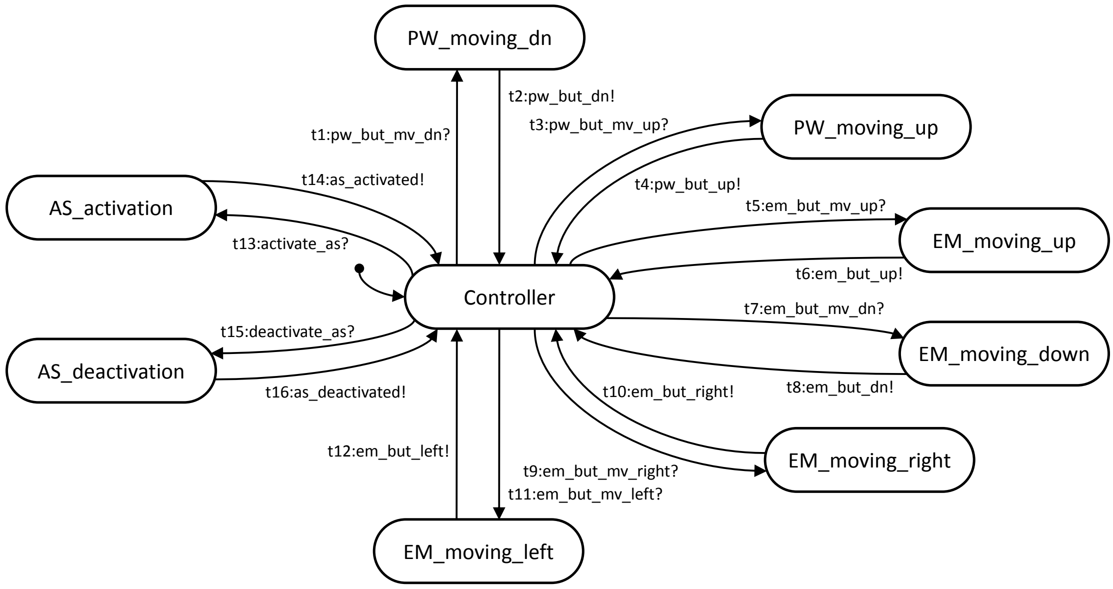

Figure 4.53.: Behavioral Specification of the Human Machine Interface Component Variant with Alarm System Feature

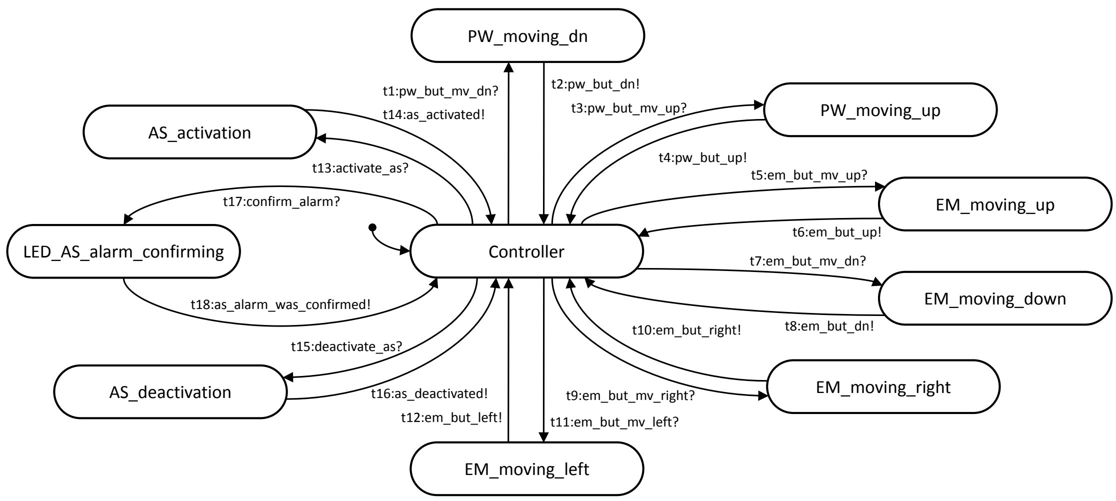

Figure 4.54.: Behavioral Specification of the Human Machine Interface Component Variant with Alarm System and LED Alarm System Features

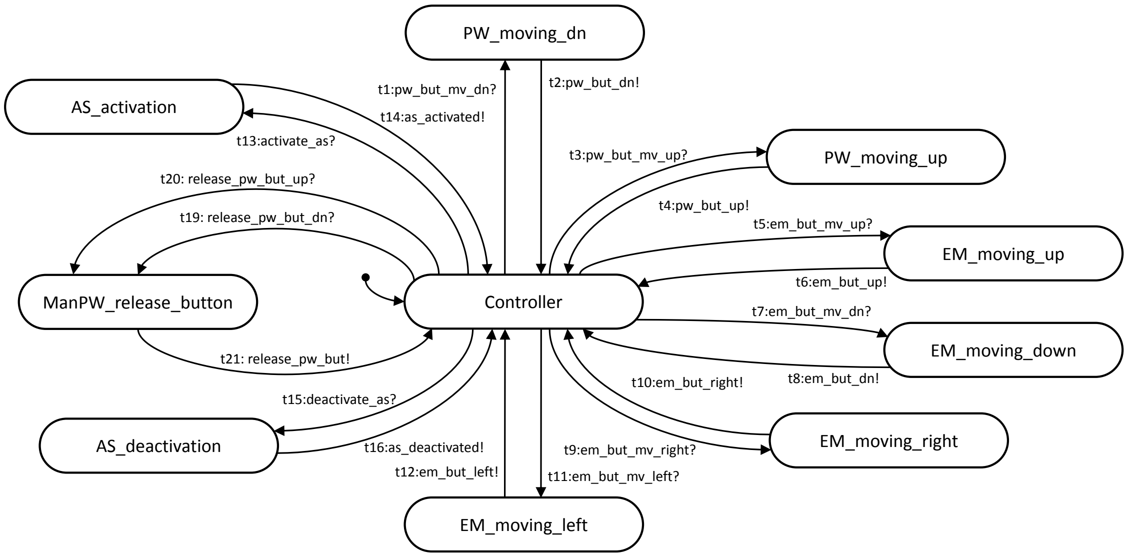

Figure 4.55.: Behavioral Specification of the Human Machine Interface Component Variant with Alarm System, Manual Power Window and LED Power Window Features

ManPW to the corresponding core. In addition to the core functionality described above, the test model variant specifies further the provision of the activation/deactivation of the alarm system, the confirmation of the silent alarm as well as the provision of the releasie state of the window buttons for the corresponding LED.

LED Automatic Power Window with Central Locking System The state machine test model variant for the component *LED Automatic Power Window* with *Central Locking System* is depicted in Fig. 4.57, comprising 13 states and 17 transitions. The test model variant is obtained by applying the delta *DAddLEDAutoPWCLS* to the core. In addition to the core functionality described above, the test model variant specifies further the turning on of another LED for the upwards movement (*led_pw_cls_up_on*) if the automatic power window moves upwards (*pw_auto_mv_up*) and the car is locked by the central locking system (*cls_lock*). The LED is turned off (*led_pw_cls_up_off*) if the car is unlocked (*cls_unlock*) or the window movement stopped (*pw_auto_mv_stop*).

Based on the architecture model and component state machine test model definitions, we describe in the next section the interaction test scenarios used as integration test model specifications for the integration testing.

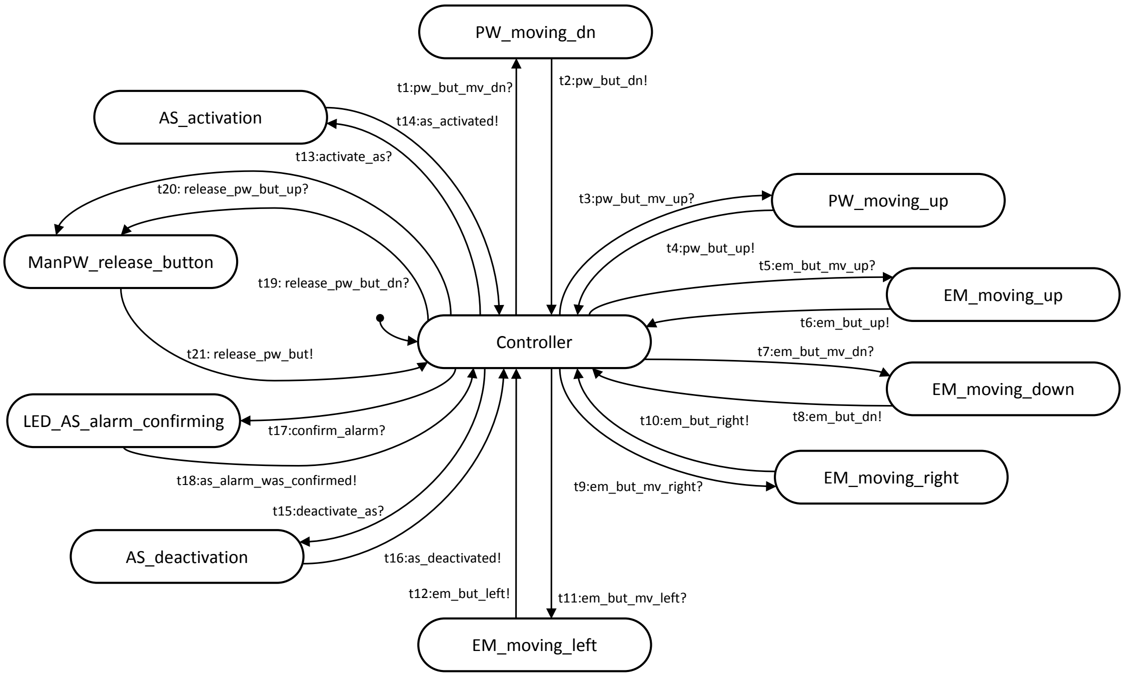

Figure 4.56.: Behavioral Specification of the Human Machine Interface Component Variant with Alarm System, LED Alarm System, Manual Power Window and LED Power Window Features

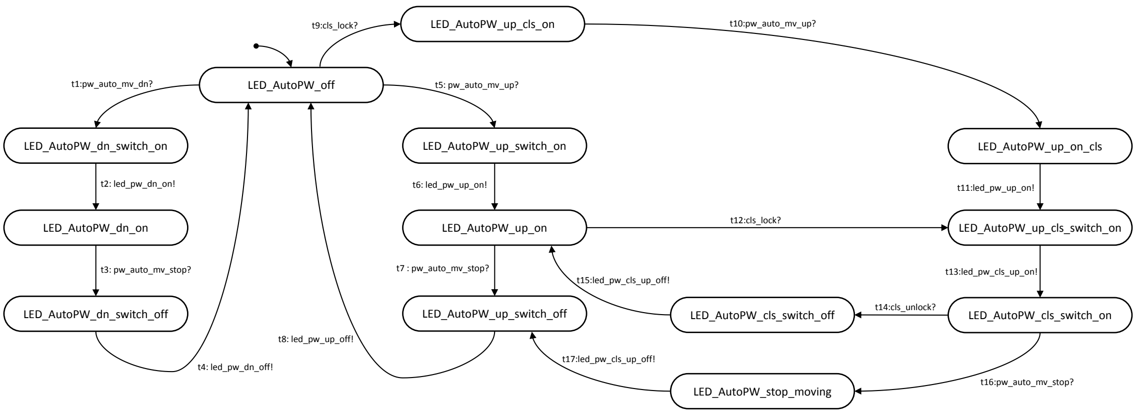

Figure 4.57.: Behavioral Specification of the LED Automatic Power Window Component Variant with Central Locking System Feature

5 BCS Integration Test Model

In this section, we document the integration test model for the BCS SPL case study. Therefore, we first describe the different message sequence charts (MSC) used for the specification of component interaction test scenarios and map each test scenario to the various product variants it is valid for. The mapping is used to deduce the integration test model variants for the representative subset as well as for the integration test model deltas.

5.1. BCS Message Sequence Charts

In this section, we describe the interaction test scenarios specified as MSCs. An MSC describes a communication scenario between different components of the system, where a component is symbolized as a box at the top of the MSC. Please note that every MSC contains an *environmental* component represented by locations labeled with l_e enabling the communication with the system environment. Components interact with each other by sending and receiving signals via the corresponding connectors represented as arrows. In total, we defined 64 interaction test scenarios as follows.

Interaction Test Scenario MSC1 The interaction test scenario *MSC1* depicted in Fig. 5.1, describes a scenario between the components *Human Machine Interface* (HMI), *Finger Protection* (FP) and *Manual Power Window* (ManPW). The scenario defines the activation of the finger protection, caused by a detected finger as well as the deactivation of the finger protection, based on the initiated downwards movement of the window.

Interaction Test Scenario MSC2 The interaction test scenario *MSC2* shown in Fig. 5.2, describes a scenario between the components *Human Machine Interface* (HMI), *Finger Protection* (FP) and *Manual Power Window* (ManPW). The scenario defines the disabling of the upwards movement of the window caused by a clamped finger. In addition, the scenario describes the re-enabling of the upwards movement, based on the release of the clamped finger by moving down the window.

Interaction Test Scenario MSC3 The interaction test scenario *MSC3* depicted in Fig. 5.3, describes a scenario between the components *Finger Protection* (FP) and *Manual Power Window* (ManPW). In contrast to *MSC2*, this scenario specifies, in addition, the reaching of the lower and upper window position.

Interaction Test Scenario MSC4 The interaction test scenario *MSC4* shown in Fig. 5.4, describes a scenario between the components *Human Machine Interface* (HMI) and *Exterior Mirror* (EM). The scenario defines the movement of the exterior mirror. The exterior is moved to its left-most, top, right-most and bottom position.

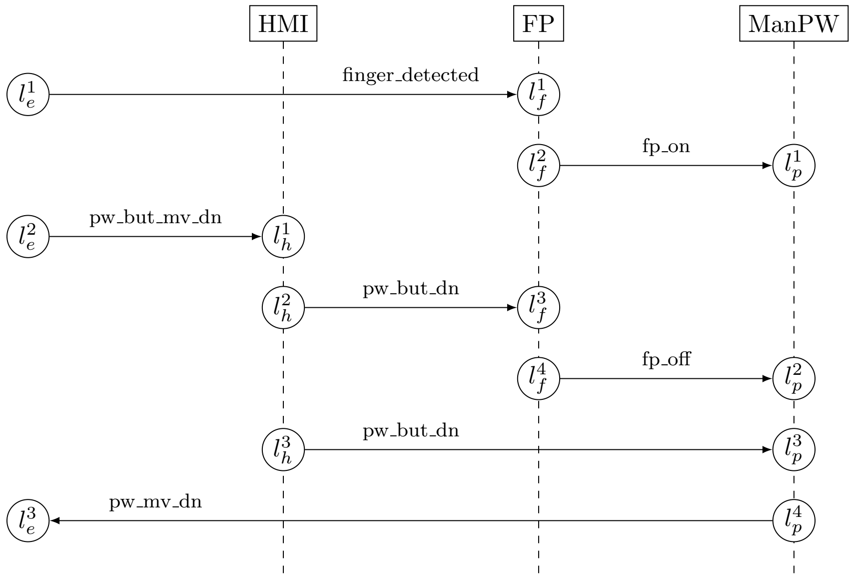

Figure 5.1.: Interaction Test Scenario MSC1

Interaction Test Scenario MSC5 The interaction test scenario *MSC5* depicted in Fig. 5.5, describes a scenario between the components *Human Machine Interface* (HMI) and *Manual Power Window* (ManPW). The scenario defines the upwards and downwards movement of the manual power window. In contrast to the previous scenarios, there is no finger protection.

Interaction Test Scenario MSC6 The interaction test scenario *MSC6* depicted in Fig. 5.6, describes a scenario between the components *Human Machine Interface* (HMI) and *Finger Protection* (FP). The scenario defines the activation and deactivation of the finger protection, caused by a clamped finger.

Interaction Test Scenario MSC7 The interaction test scenario *MSC7* shown in Fig. 5.7, describes a scenario between the components *Finger Protection* (FP) and *LED Finger Protection* (LED FP). The scenario defines the turning on and off of the finger protection LED, based on the activation/deactivation of the finger protection.

Interaction Test Scenario MSC8 The interaction test scenario *MSC8* depicted in Fig. 5.8, describes a scenario between the components *Central Locking System* (CLS) and *Automatic Power Window* (AutoPW). The scenario defines the locking of the car and the blocking of the downwards movement of the window if the car is locked.

Interaction Test Scenario MSC9 The interaction test scenario *MSC9* shown in Fig. 5.9, describes a scenario between the components *Finger Protection* (FP) and *Automatic Power Window* (AutoPW). The scenario defines the activation of the finger protection and the blocking of the automated upwards movement of the window, based on the activated finger protection. In addition, the scenario describes the stopping of the automated downwards mo-

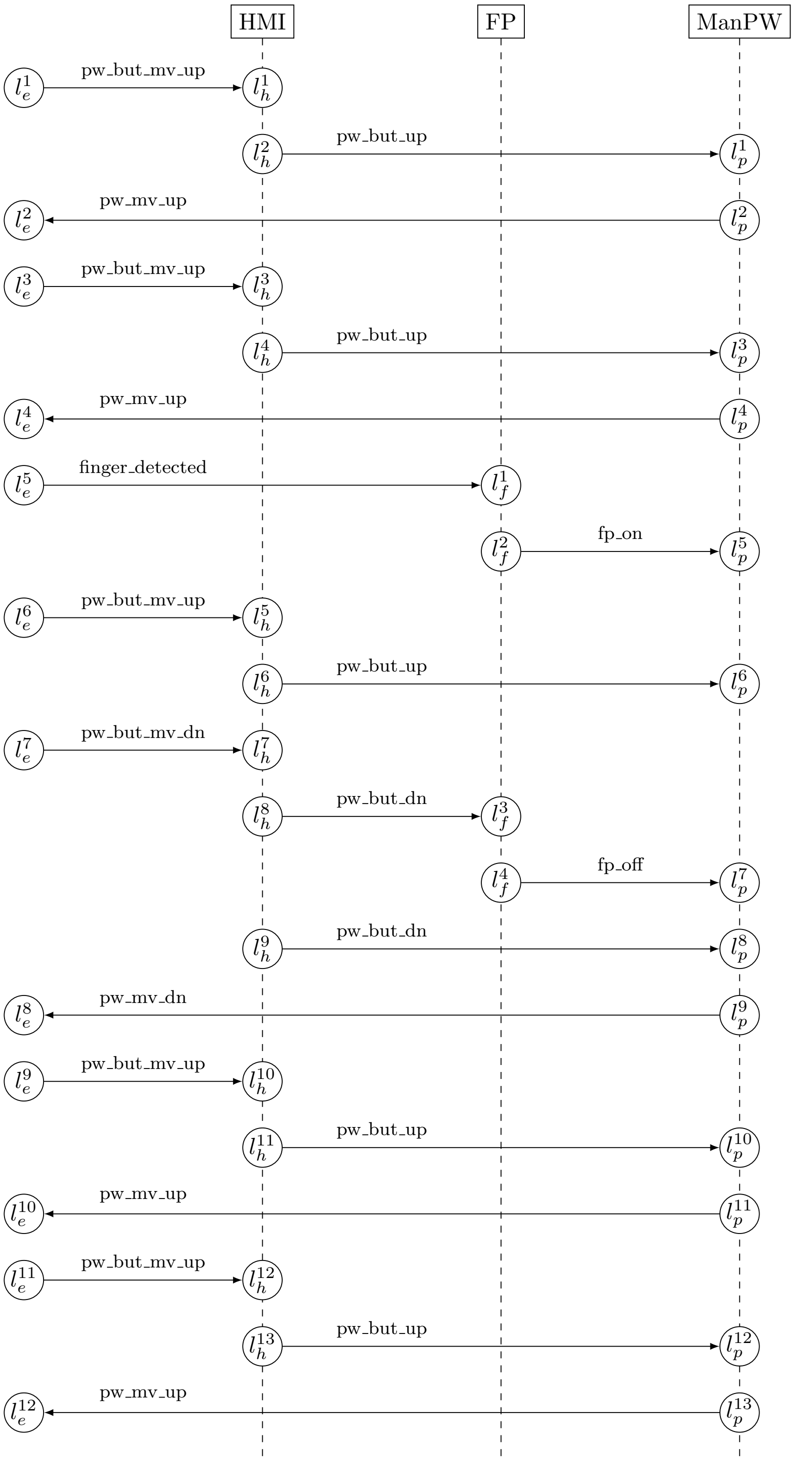

Figure 5.2.: Interaction Test Scenario MSC2

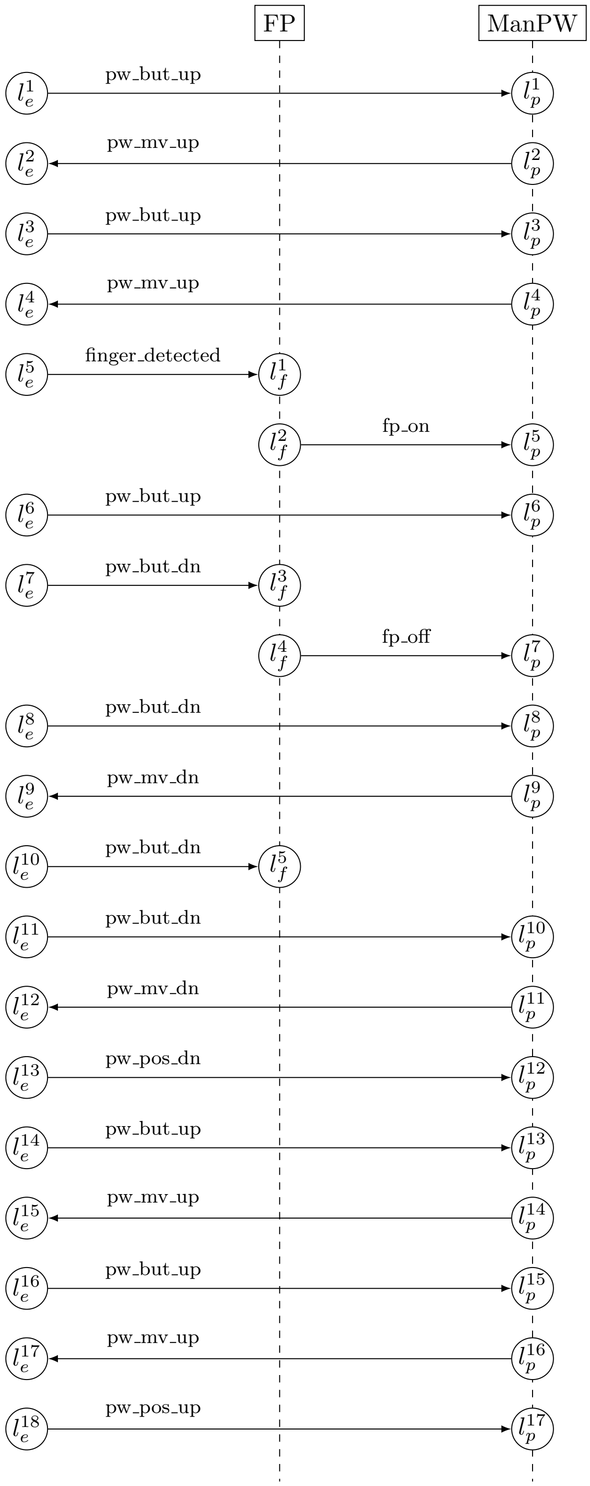

Figure 5.3.: Interaction Test Scenario MSC3

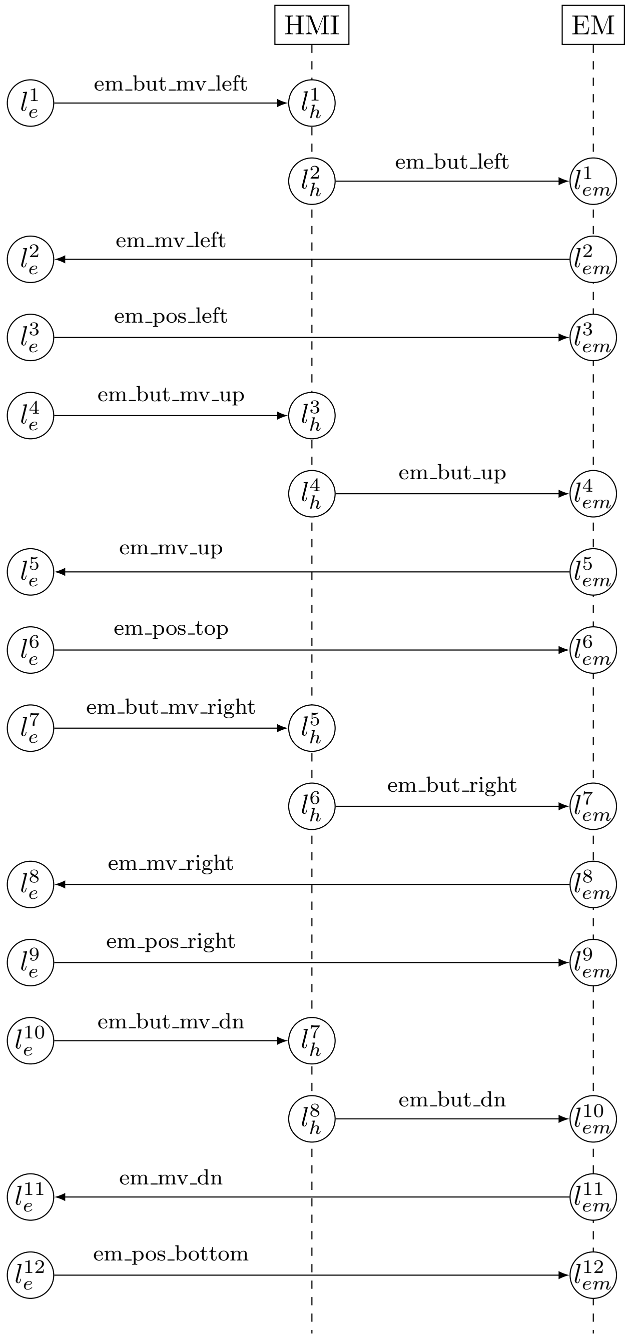

Figure 5.4.: Interaction Test Scenario MSC4

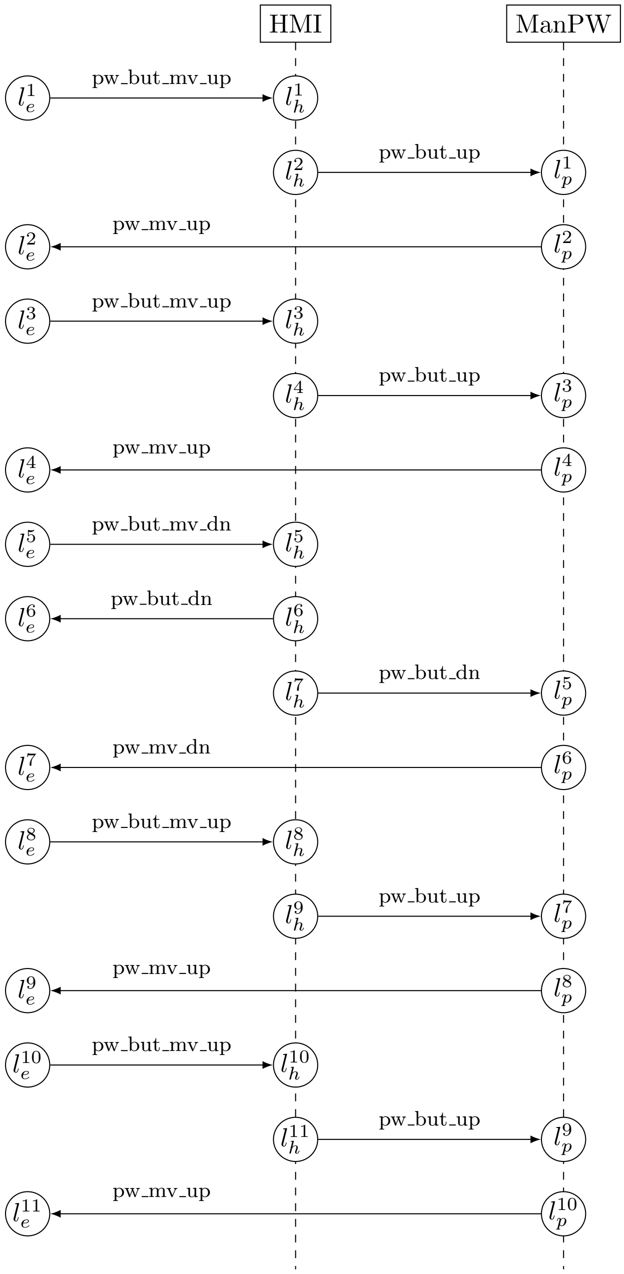

Figure 5.5.: Interaction Test Scenario MSC5

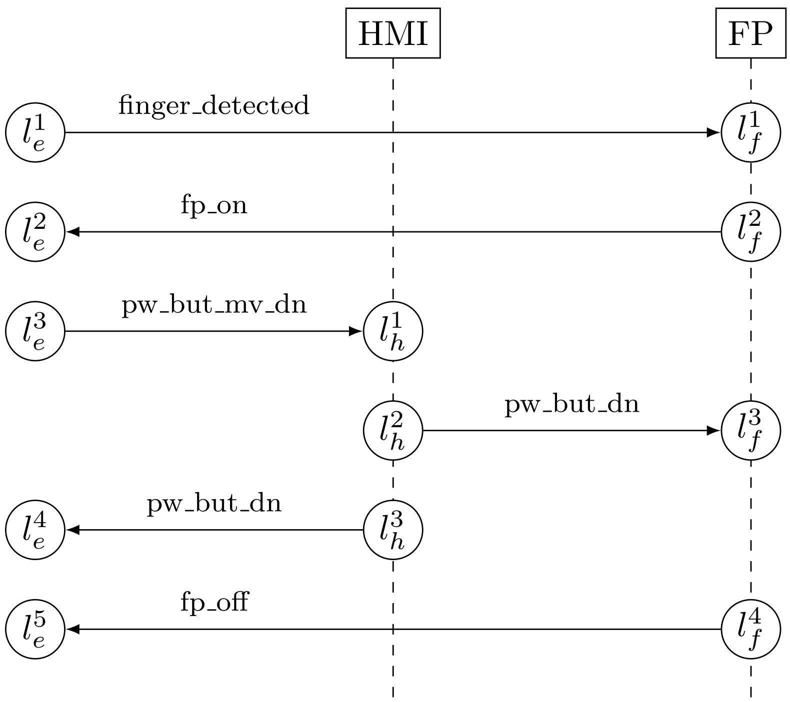

Figure 5.6.: Interaction Test Scenario MSC6

vement if the window reaches its lower position.

Interaction Test Scenario MSC10 The interaction test scenario *MSC10* describes a scenario between the components *Central Locking System* (CLS), *Finger Protection* (FP) and *Automatic Power Window* (AutoPW). It is depicted in Fig. 5.10 and Fig. 5.11, where the dashed horizontal line represents a cut for better readability. The scenario defines the activation of the finger protection and the blocking of the upwards movement of the window, based on the locking of the car. In addition, the scenario describes the locking and unlocking of the car, based on the activated/deactivated central locking system.

Interaction Test Scenario MSC11 The interaction test scenario *MSC11* shown in Fig. 5.12, describes a scenario between the components *Human Machine Interface* (HMI), *Central Locking System* (CLS) and *Automatic Power Window* (AutoPW). The scenario defines the locking of the car and the blocking of the downwards movement of the window, based on the activated central locking system.

Interaction Test Scenario MSC12 The interaction test scenario *MSC12* depicted in Fig. 5.13, defines a scenario between the *Human Machine Interface* (HMI), *Finger Protection* (FP) and *LED Finger Protection* (LED FP). The scenario describes the turning on and off of the finger protection LED, based on the activation/deactivation of the finger protection.

Interaction Test Scenario MSC13 The interaction test scenario *MSC13* defines a scenario between the *Human Machine Interface* (HMI), *Central Locking System* (CLS), *Finger Protection* (FP)

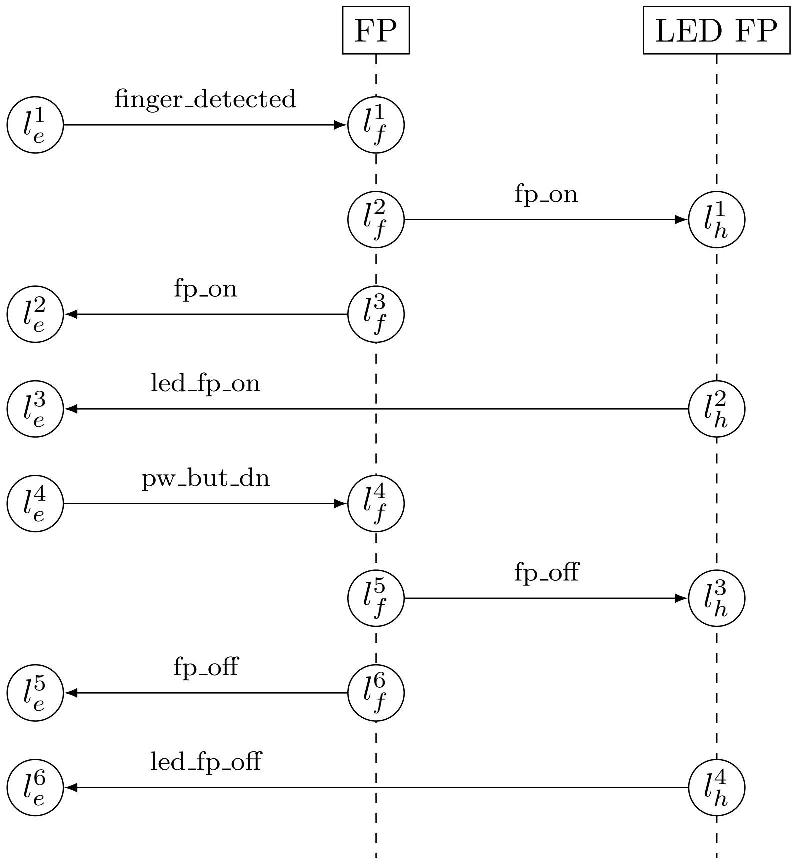

Figure 5.7.: Interaction Test Scenario MSC7

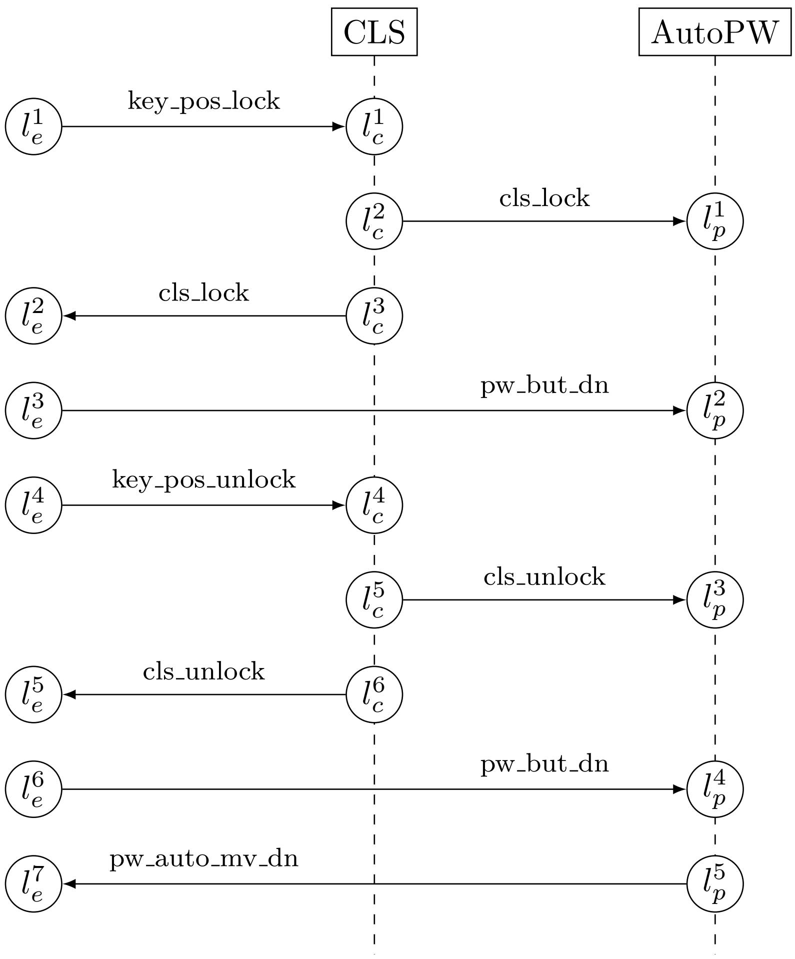

Figure 5.8.: Interaction Test Scenario MSC8

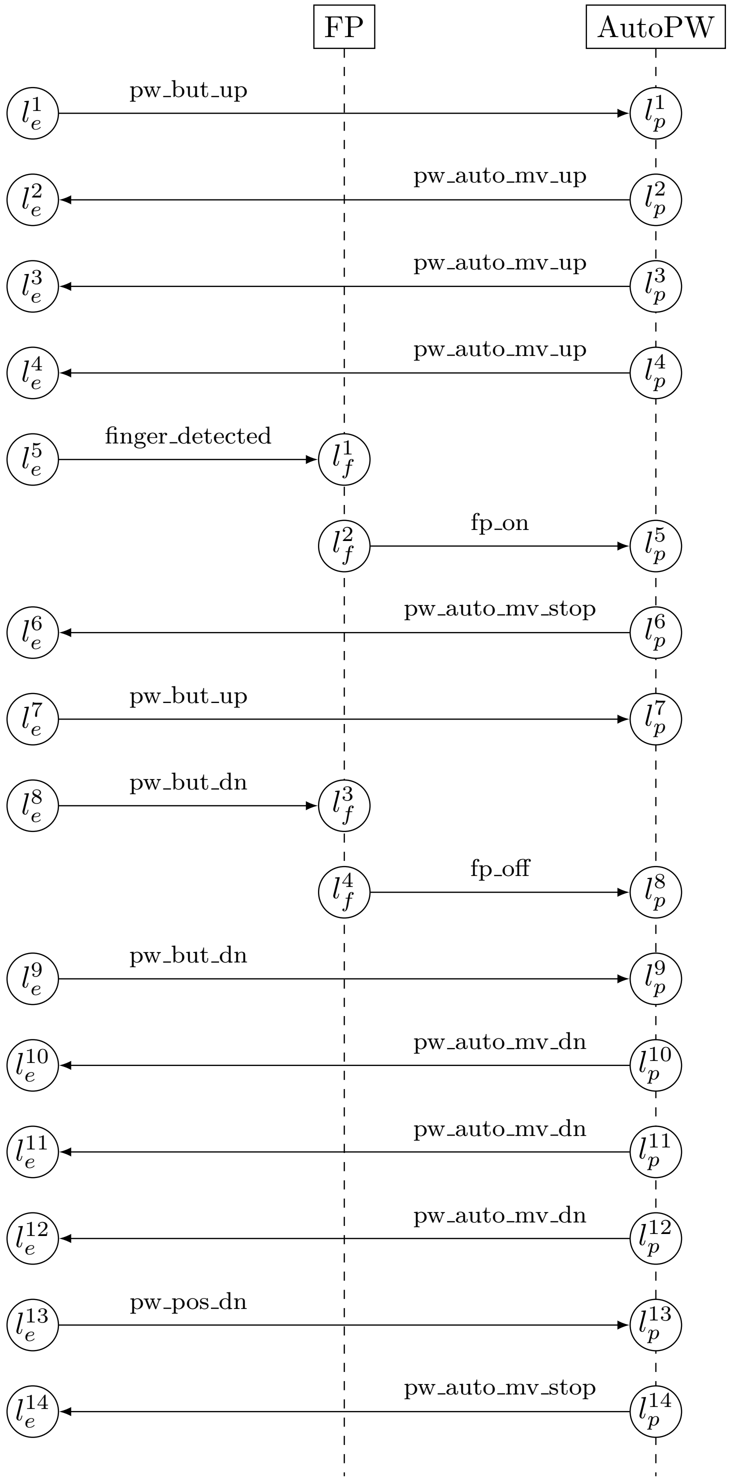

Figure 5.9.: Interaction Test Scenario MSC9

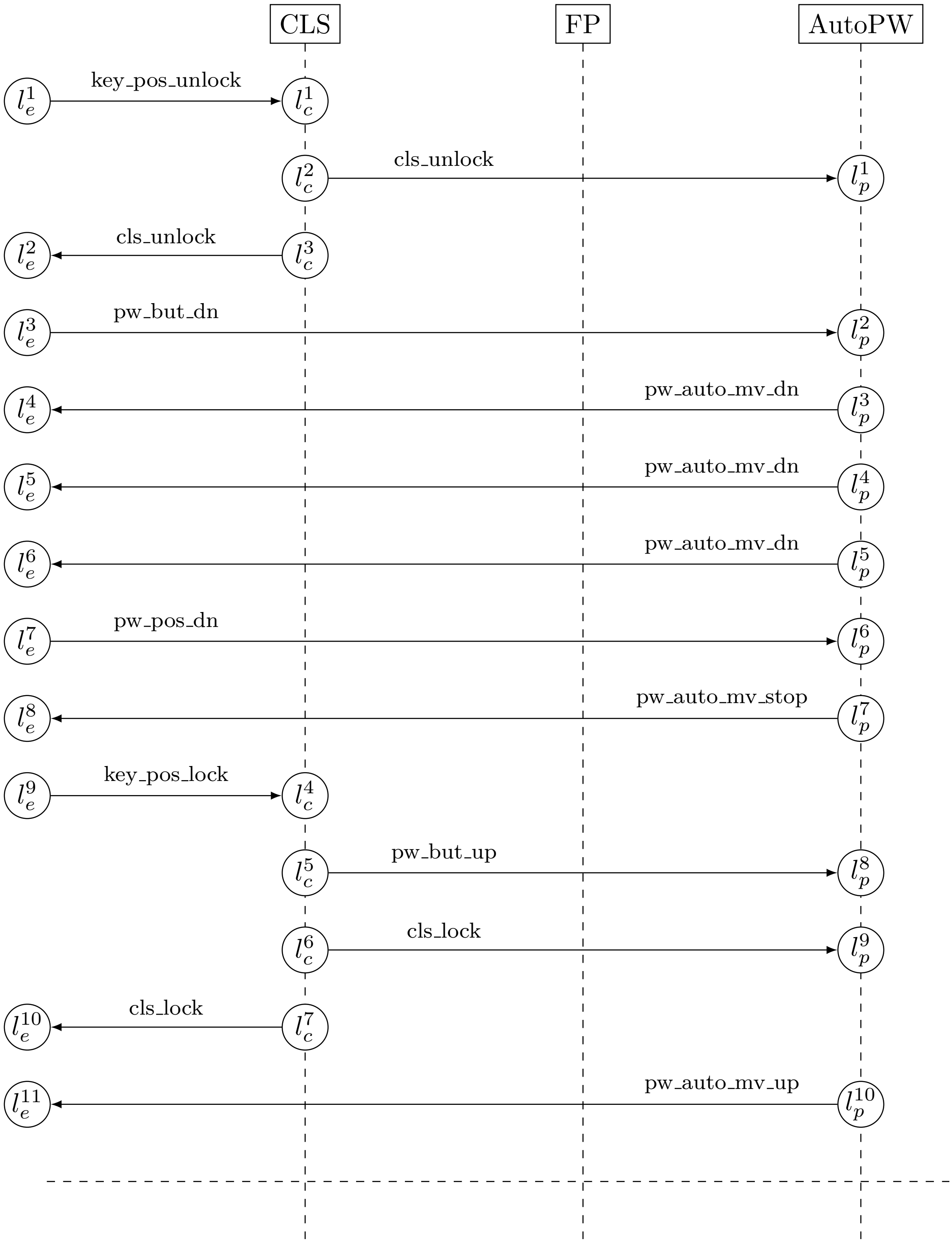

Figure 5.10.: Interaction Test Scenario MSC10 (1)

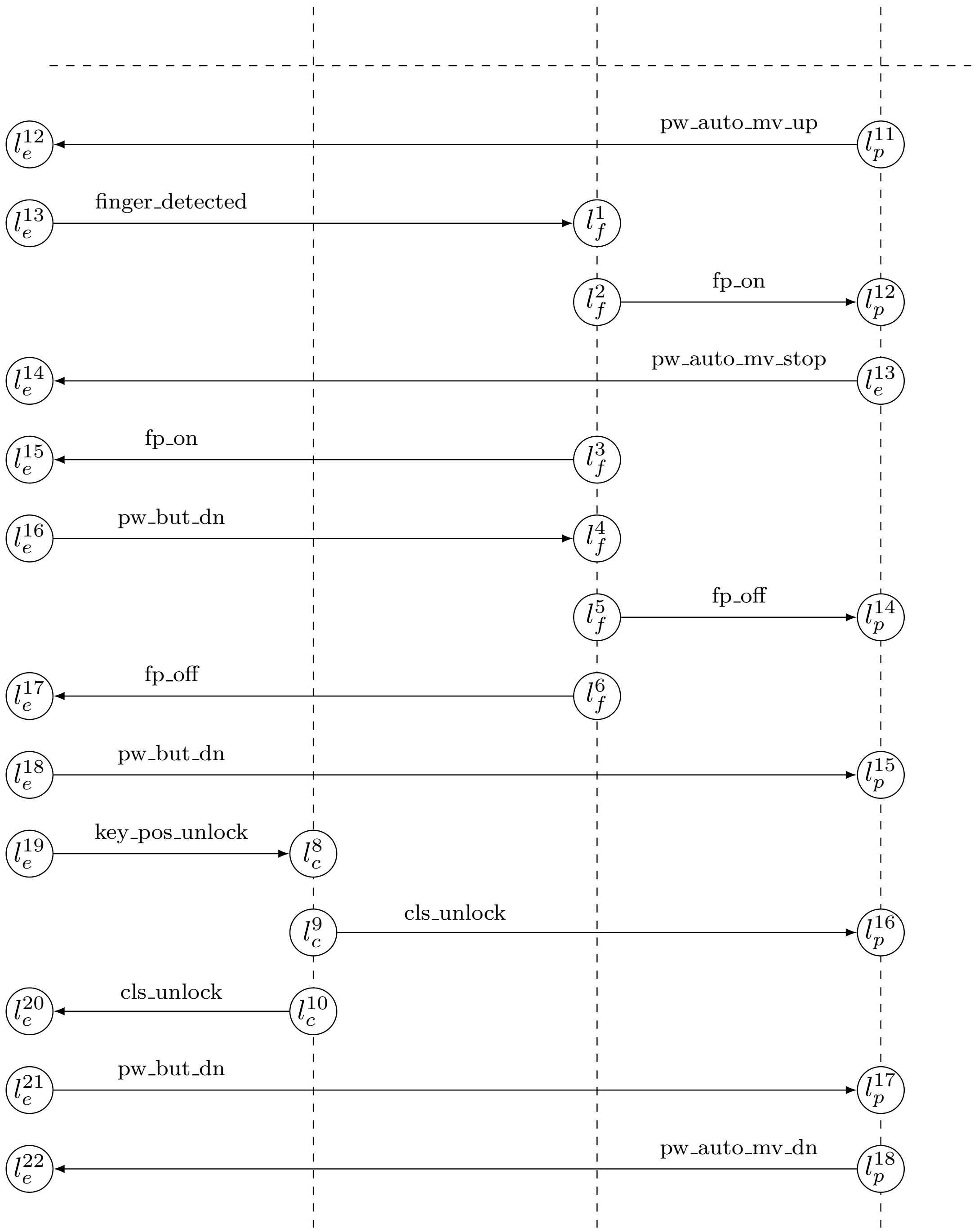

Figure 5.11.: Interaction Test Scenario MSC10 (2)

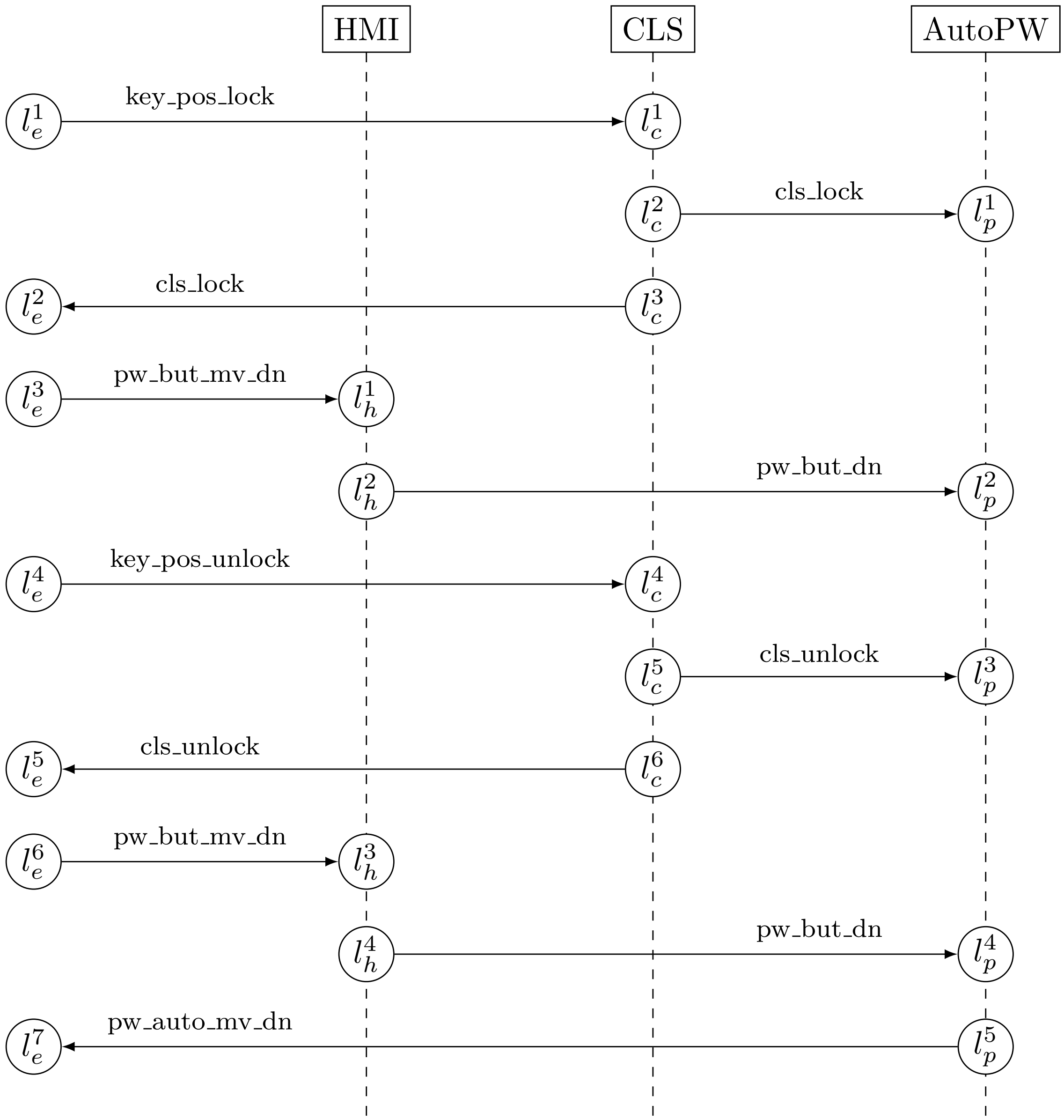

Figure 5.12.: Interaction Test Scenario MSC11

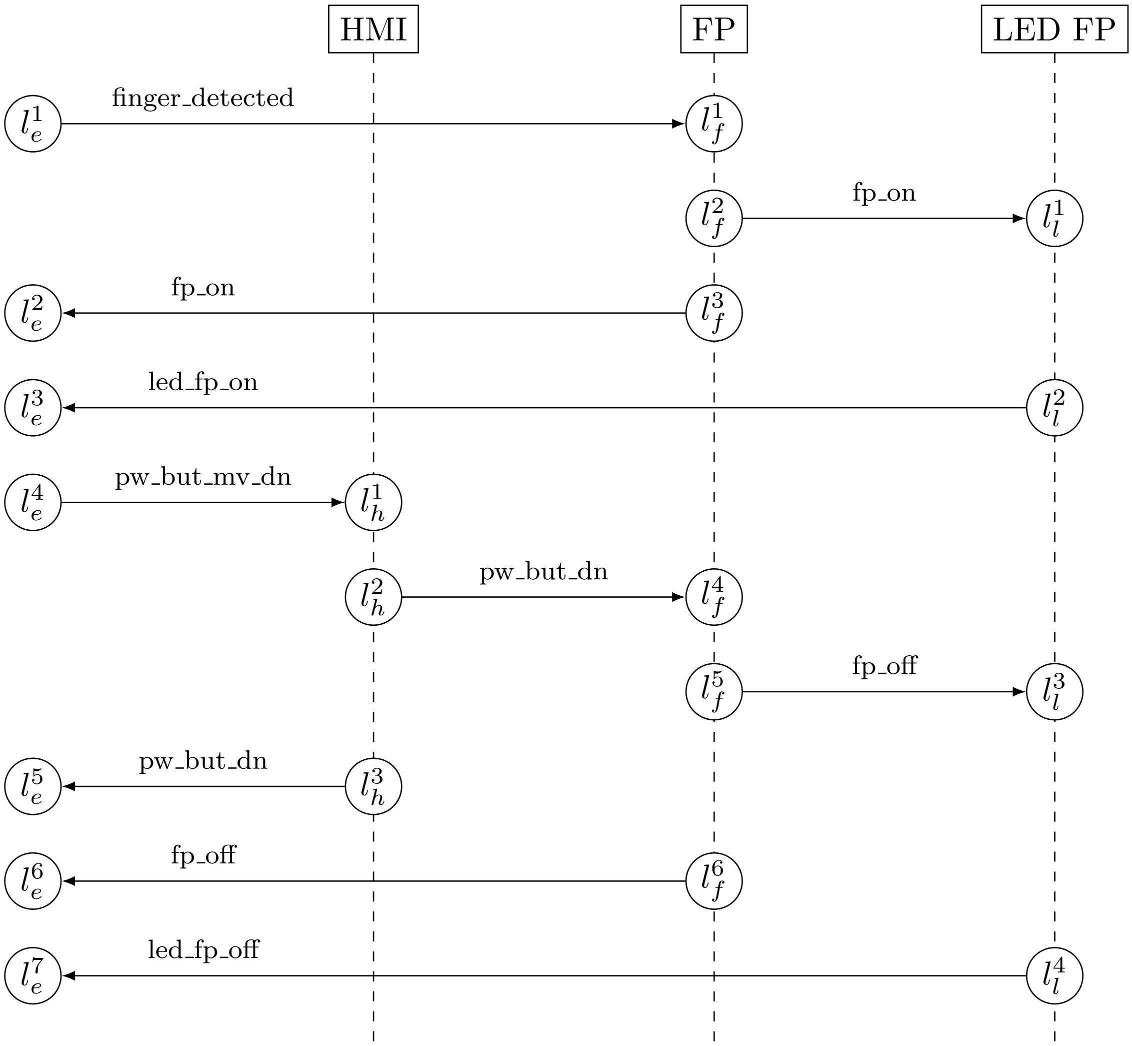

Figure 5.13.: Interaction Test Scenario MSC12

and *Automatic Power Window* (AutoPW). It is shown in Fig. 5.14 and Fig. 5.15, where the dashed horizontal line represents a cut for better readability. The scenario describes the downwards movement and the stopping of the automated window movement as well as its upwards movement, based on the locking of the car. In addition, the activation and deactivation of the finger protection is defined.

Interaction Test Scenario MSC14 The interaction test scenario *MSC14* depicted in Fig. 5.16, defines a scenario between the *Human Machine Interface* (HMI), *Finger Protection* (FP) and *Automatic Power Window* (AutoPW). The scenario describes the activation and deactivation of the finger protection, based on the corresponding upwards/downwards movement of the window.

Interaction Test Scenario MSC15 The interaction test scenario *MSC15* defines a scenario between the *Human Machine Interface* (HMI) and *Exterior Mirror with Heating* (EMH). It is shown in Fig. 5.17 and Fig. 5.18, where the dashed horizontal line represents a cut for better readability. The scenario describes the movement of the exterior mirror. The exterior mirror is moved to its left-most, top, right-most and bottom position. In addition, the activation and deactivation of the mirror heater is defined.

Interaction Test Scenario MSC16 The interaction test scenario *MSC16* defines a scenario between the *Remote Control Key Controller* (RCK Ctrl), the *Central Locking System* (CLS), the *Finger Protection* (FP) and the *Exterior Mirror with Heating* (EMH). It is depicted in Fig. 5.19 and Fig. 5.20, where the dashed horizontal line represents a cut for better readability. The scenario describes the unlocking and locking of the car using the remote key. In addition, the scenario defines the movement of the power window as well as its automated upwards movement, based on the locking of the car using the remote key. Furthermore, the activation and deactivation of the finger protection is specified.

Interaction Test Scenario MSC17 The interaction test scenario *MSC17* shown in Fig. 5.21, defines a scenario between the *Remote Control Key Controller* (RCK Ctrl) and the *Central Locking System* (CLS). The scenario describes the locking and unlocking of the car, based on the activated/deactivated central locking system controlled by the remote key.

Interaction Test Scenario MSC18 The interaction test scenario *MSC18* defines a scenario between the *Human Machine Interface* (HMI), the *Remote Control Key Controller* (RCK Ctrl), the *Central Locking System* (CLS), the *Finger Protection* (FP) and the *Automatic Power Window* (AutoPW). It is depicted in Fig. 5.22 and Fig. 5.23, where the dashed horizontal line represents a cut for better readability. The scenario describes the locking and unlocking of the car, based on the activated/deactivated central locking system controlled by the remote key. In addition, the scenario defines the automated upwards movement of the window caused by the locking of the car as well as the activation and deactivation of the finger protection, caused by a clamped finger.

Interaction Test Scenario MSC19 The interaction test scenario *MSC19* shown in Fig. 5.24, defines a scenario between the *Human Machine Interface* (HMI) and the *Alarm System* (AS). The

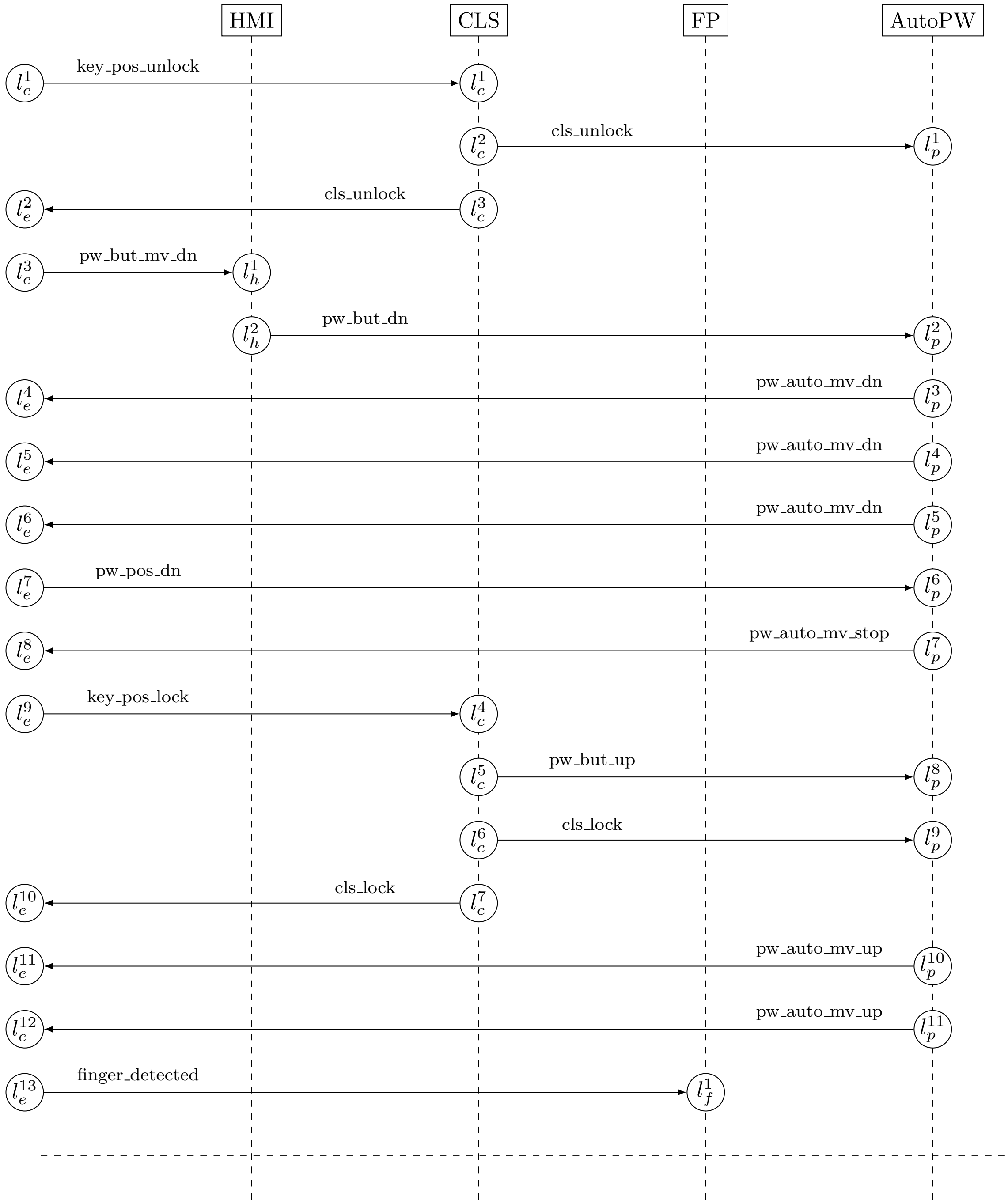

Figure 5.14.: Interaction Test Scenario MSC13 (1)

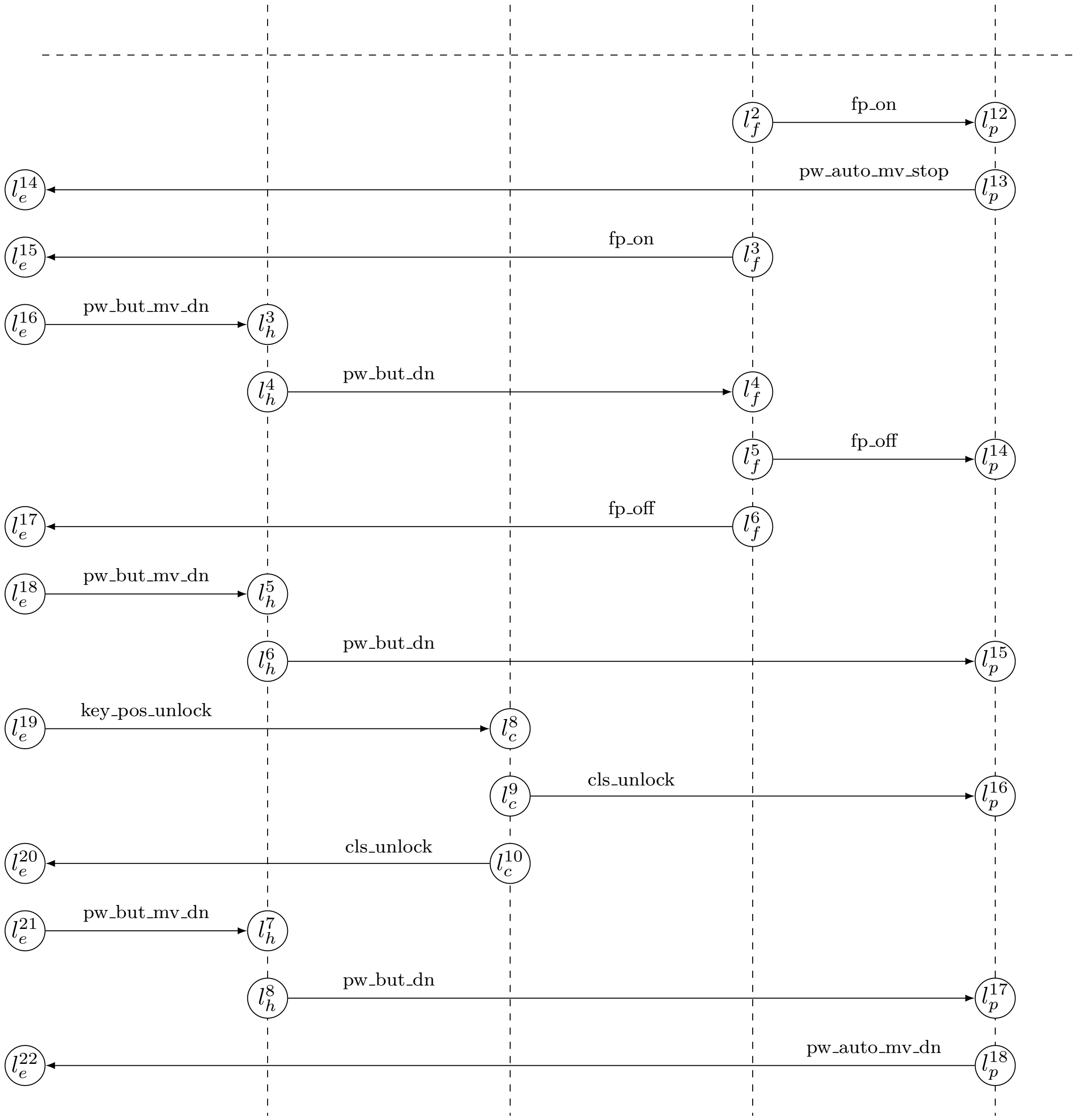

Figure 5.15.: Interaction Test Scenario MSC13 (2)

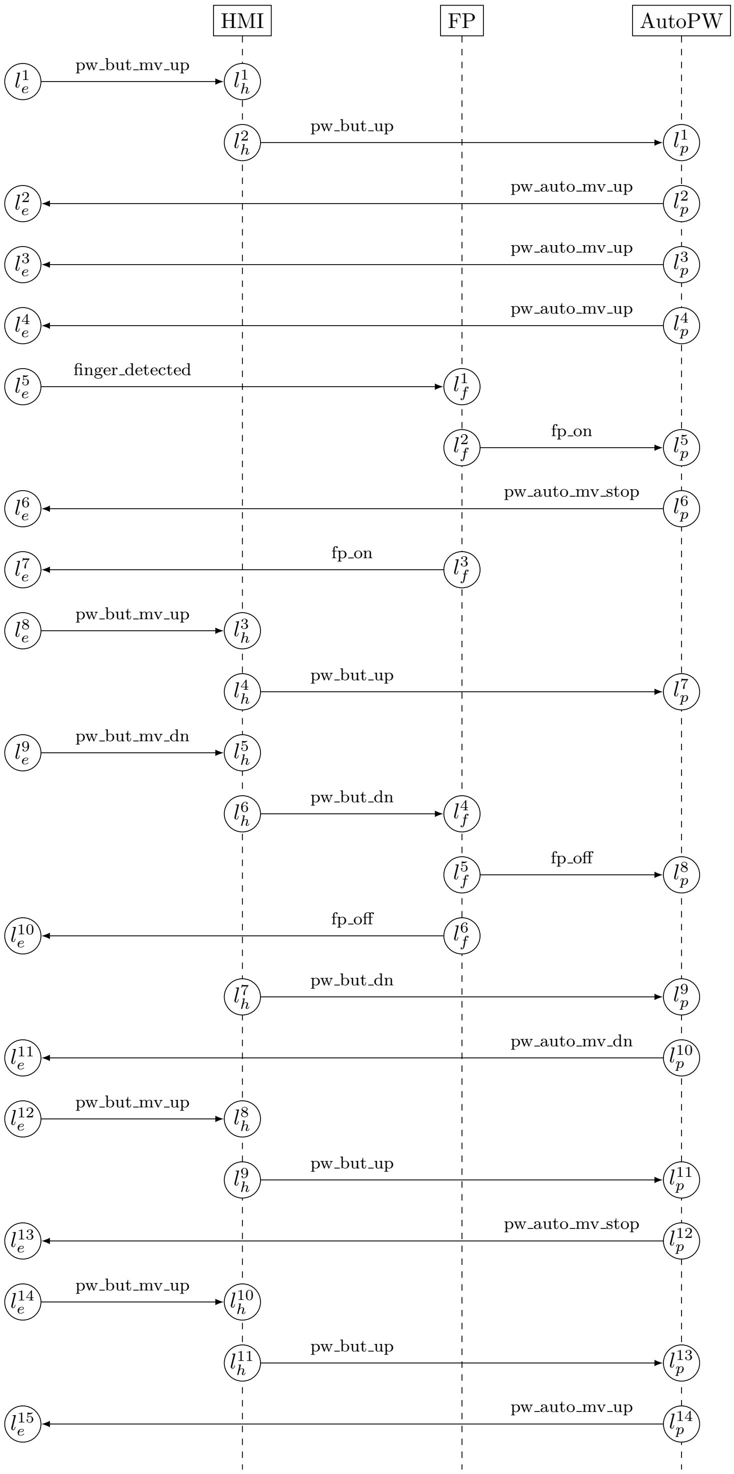

Figure 5.16.: Interaction Test Scenario MSC14

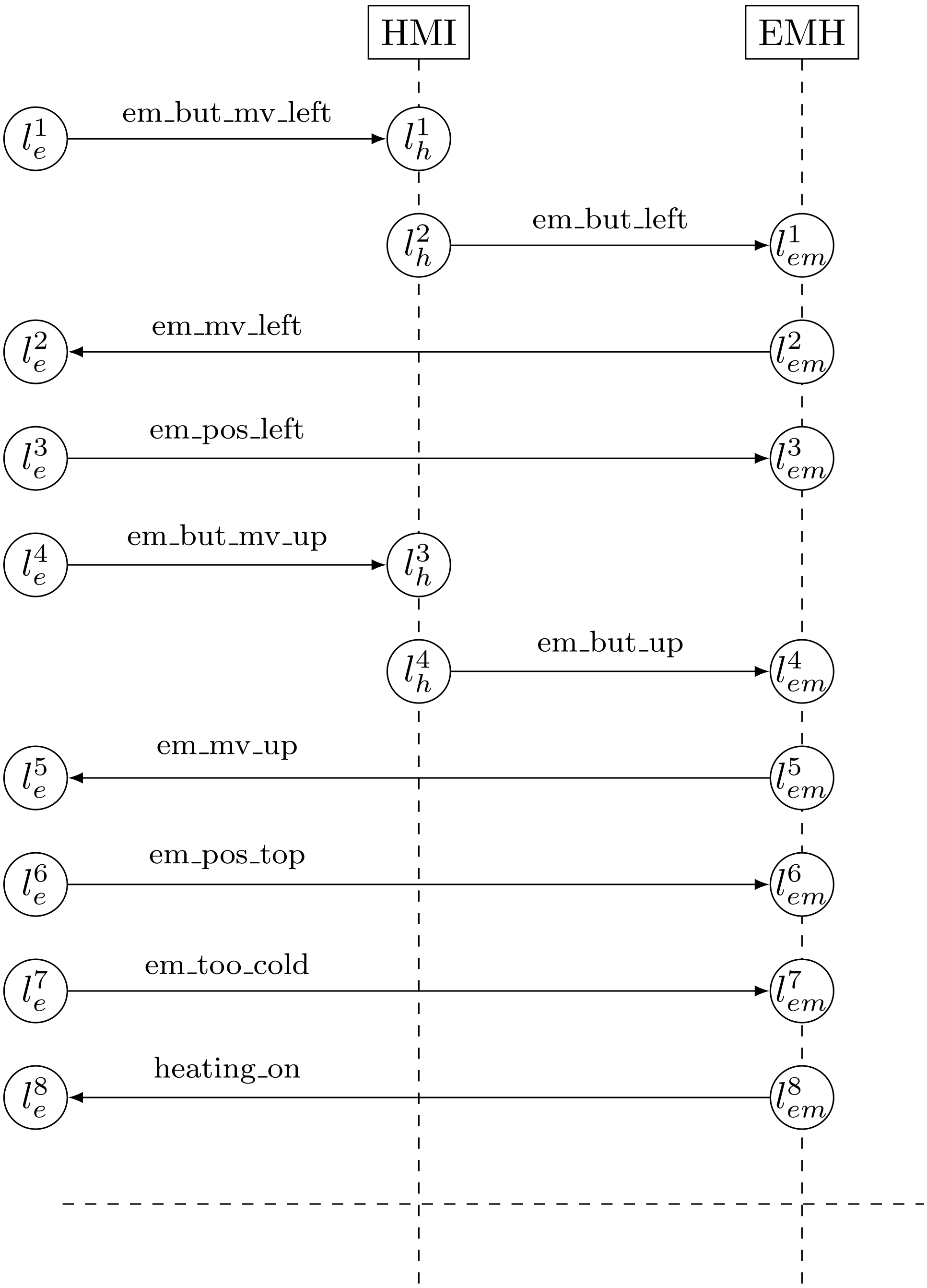

Figure 5.17.: Interaction Test Scenario MSC15 (1)

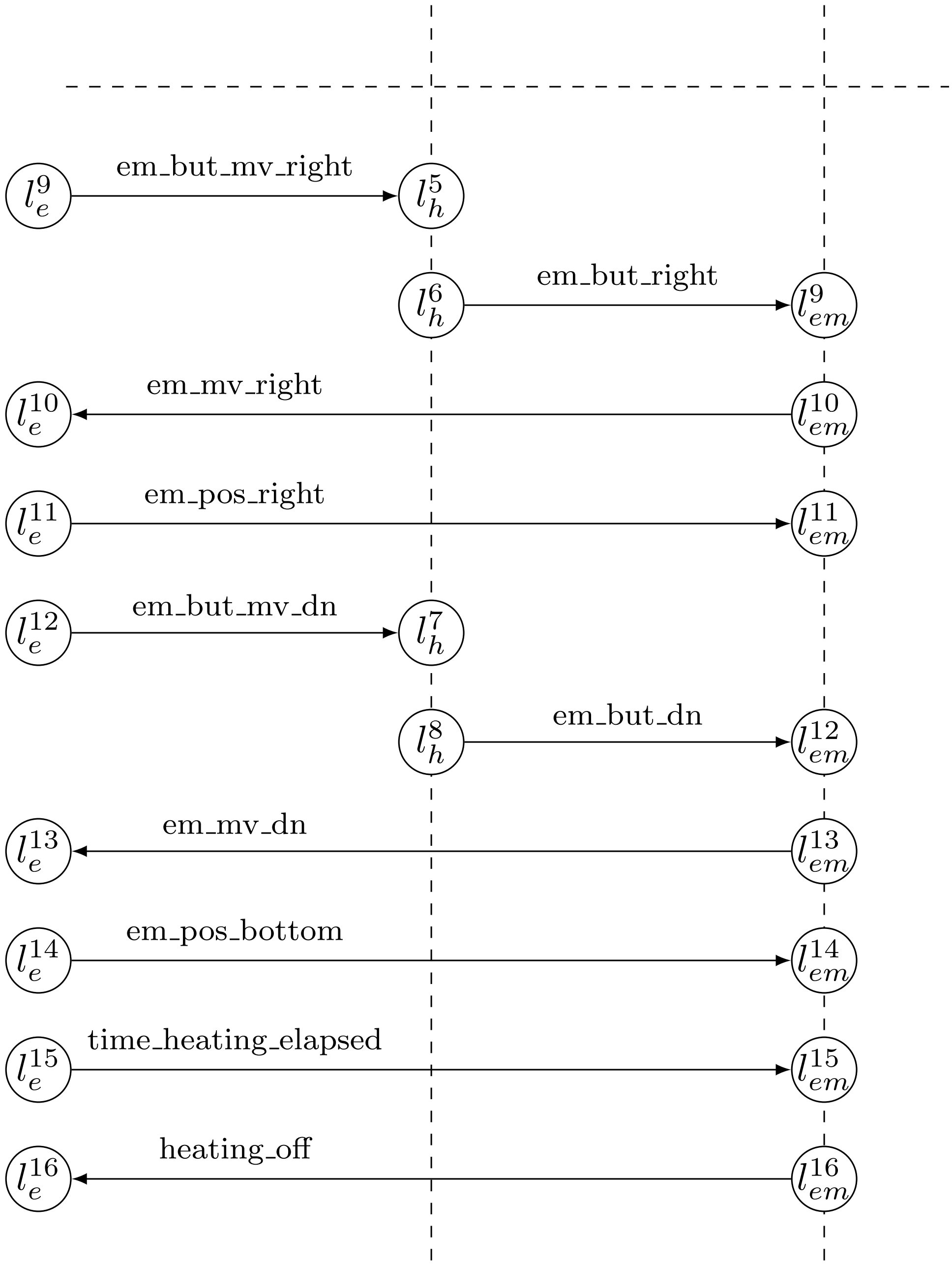

Figure 5.18.: Interaction Test Scenario MSC15 (2)

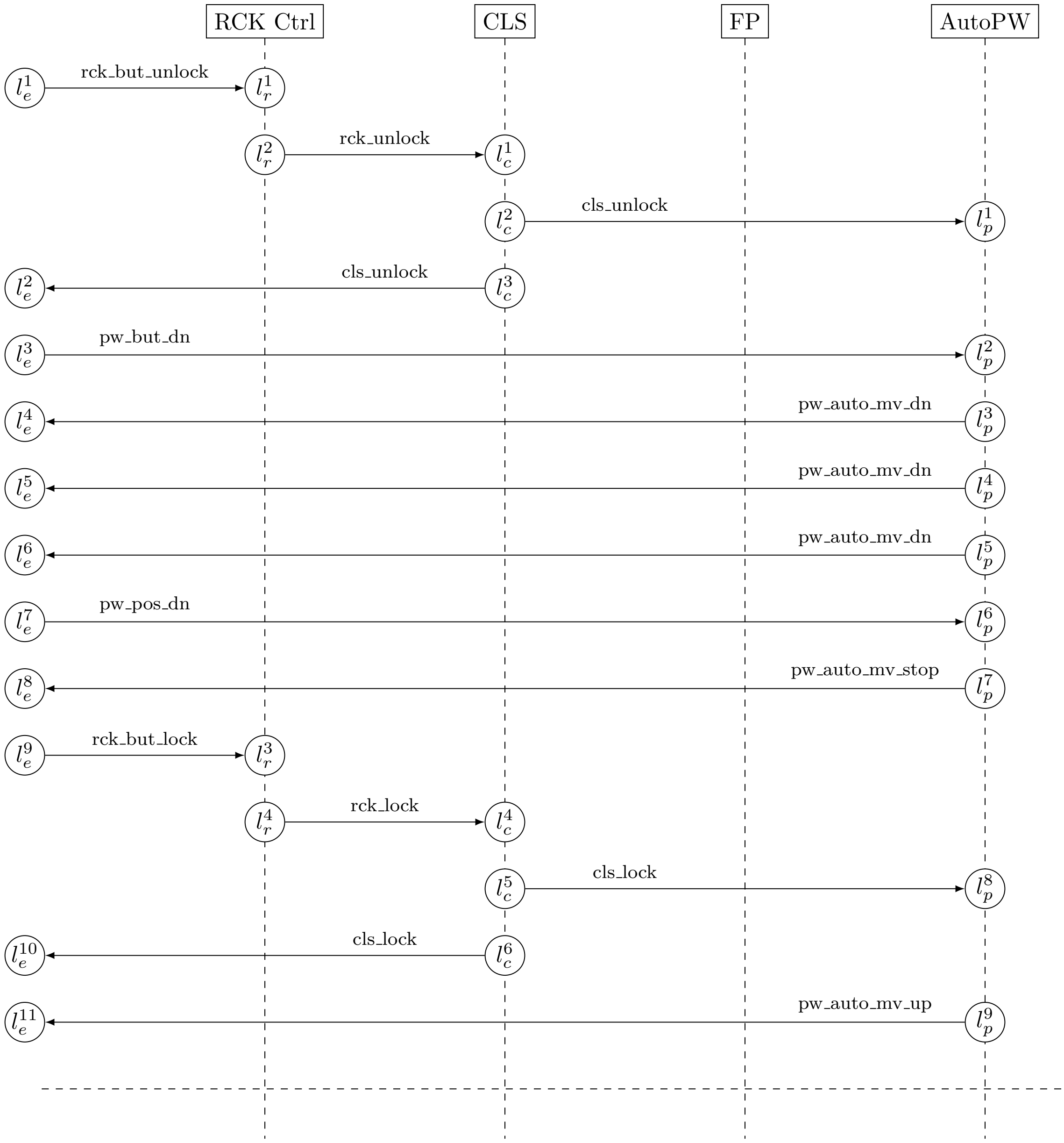

Figure 5.19.: Interaction Test Scenario MSC16 (1)

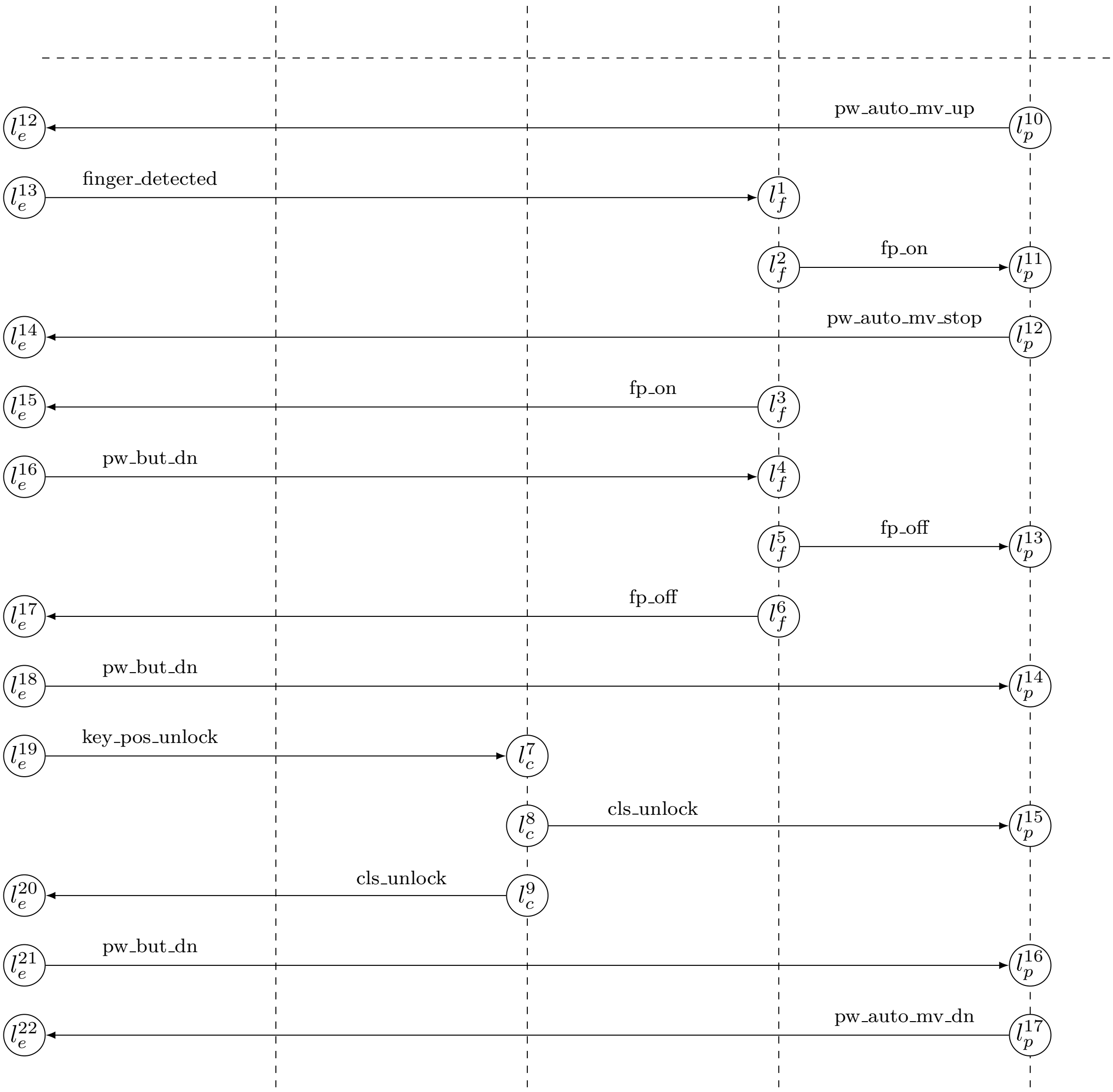

Figure 5.20.: Interaction Test Scenario MSC16 (2)

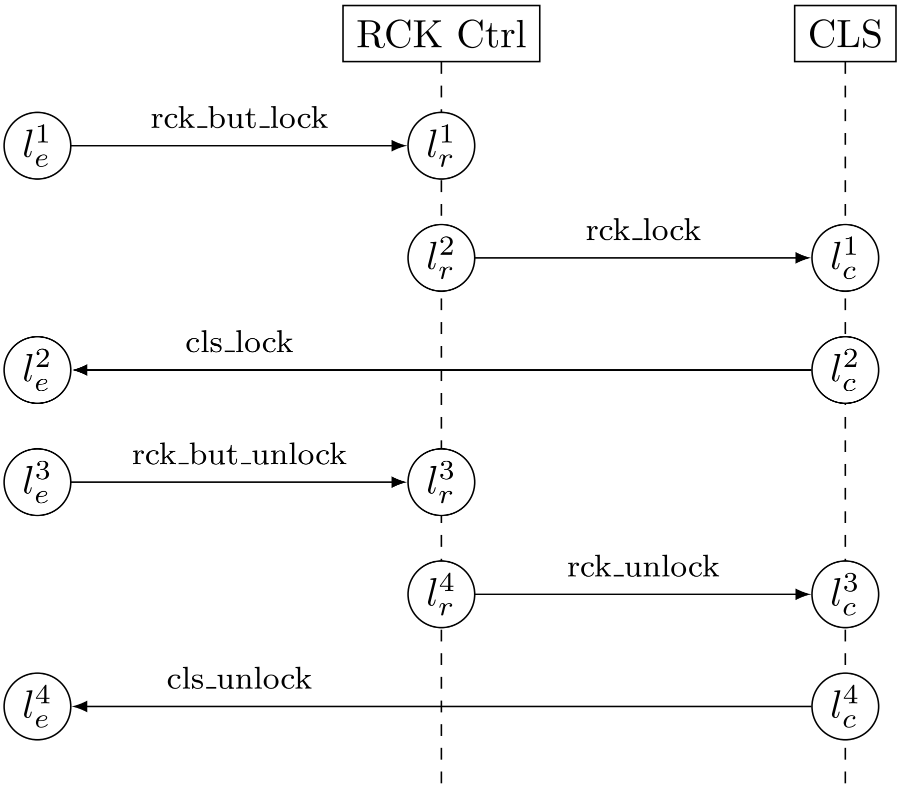

Figure 5.21.: Interaction Test Scenario MSC17

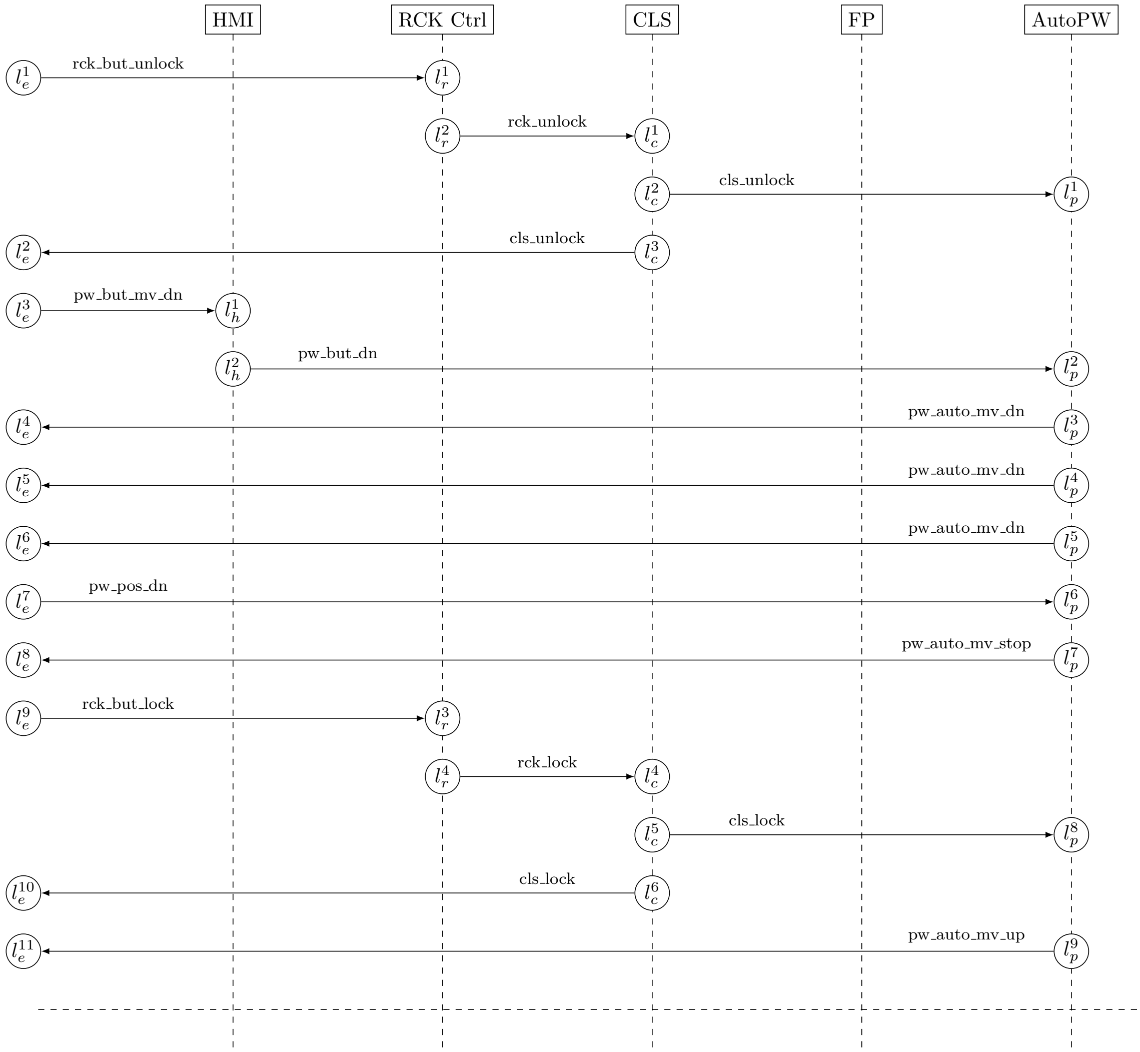

Figure 5.22.: Interaction Test Scenario MSC18 (1)

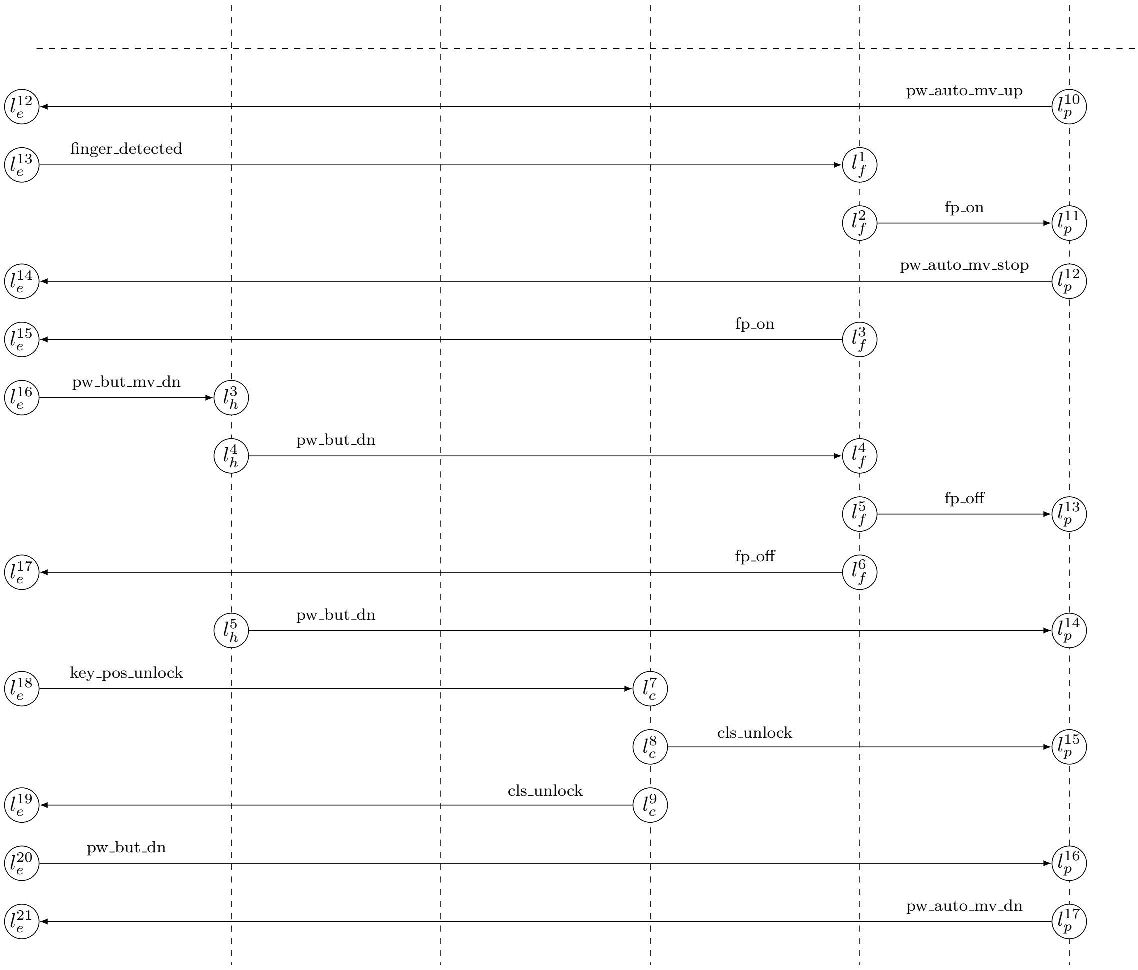

Figure 5.23.: Interaction Test Scenario MSC18 (2)

scenario describes the activation and deactivation of the alarm system, based on the interaction of the driver with the human machine interface. In addition, the scenario defines the triggering of the alarm and its deactivation.

Interaction Test Scenario MSC20 The interaction test scenario *MSC20* depicted in Fig. 5.25, defines a scenario between the *Human Machine Interface* (HMI) and the *Alarm System* (AS). The scenario describes the activation and deactivation of the alarm system, based on the interaction of the driver with the human machine interface. In addition, the scenario defines the triggering of the interior alarm and its deactivation.

Interaction Test Scenario MSC21 The interaction test scenario *MSC21* shown in Fig. 5.26, defines a scenario between the *Human Machine Interface* (HMI) and the *Alarm System* (AS). The scenario describes the activation of the alarm system, based on the interaction of the driver with the human machine interface. In addition, the scenario defines the triggering and the deactivation of the alarm as well as the disabling of the alarm monitoring of the alarm system by unlocking the car.

Interaction Test Scenario MSC22 The interaction test scenario *MSC22* depicted in Fig. 5.27, defines a scenario between the *Human Machine Interface* (HMI) and the *Alarm System* (AS). The scenario describes the activation of the alarm system as well as the enabling of the alarm monitoring, based on the locking of the car. In addition, the scenario defines the triggering of the alarm and the sending of the silent alarm after the alarm time elapsed. The scenario ends with the disabling of the alarm monitoring and the deactivation of the alarm system.

Interaction Test Scenario MSC23 The interaction test scenario *MSC23* shown in Fig. 5.28, defines a scenario between the *Finger Protection* (FP), the *LED Finger Protection* (LED FP) and the *Automatic Power Window* (AutoPW). The scenario describes the activation of the finger protection resulting in the turning on of the LED and the stopping of the automated power window. In addition, the scenario defines the deactivation of the finger protection and the release of the power window by moving it downwards.

Interaction Test Scenario MSC24 The interaction test scenario *MSC24* depicted in Fig. 5.29, describes a scenario between the *Human Machine Interface* (HMI) and the *Automatic Power Window* (AutoPW). The scenario defines the upwards movement of the automatic power window as well as its stopping by pressing the corresponding button on the human machine interface.

Interaction Test Scenario MSC25 The interaction test scenario *MSC25* shown in Fig. 5.30, describes a scenario between the *Remote Control Key Controller* (RCK Ctrl) and the *Alarm System* (AS). The scenario defines the disabling and enabling of the alarm monitoring of the activated alarm system, caused by unlocking and locking the car using the remote key.

Interaction Test Scenario MSC26 The interaction test scenario *MSC26* depicted in Fig. 5.31, describes a scenario between the *Remote Control Key Controller* (RCK Ctrl) and the *Alarm System* (AS). The scenario defines the activation of the alarm system, based on the locking of

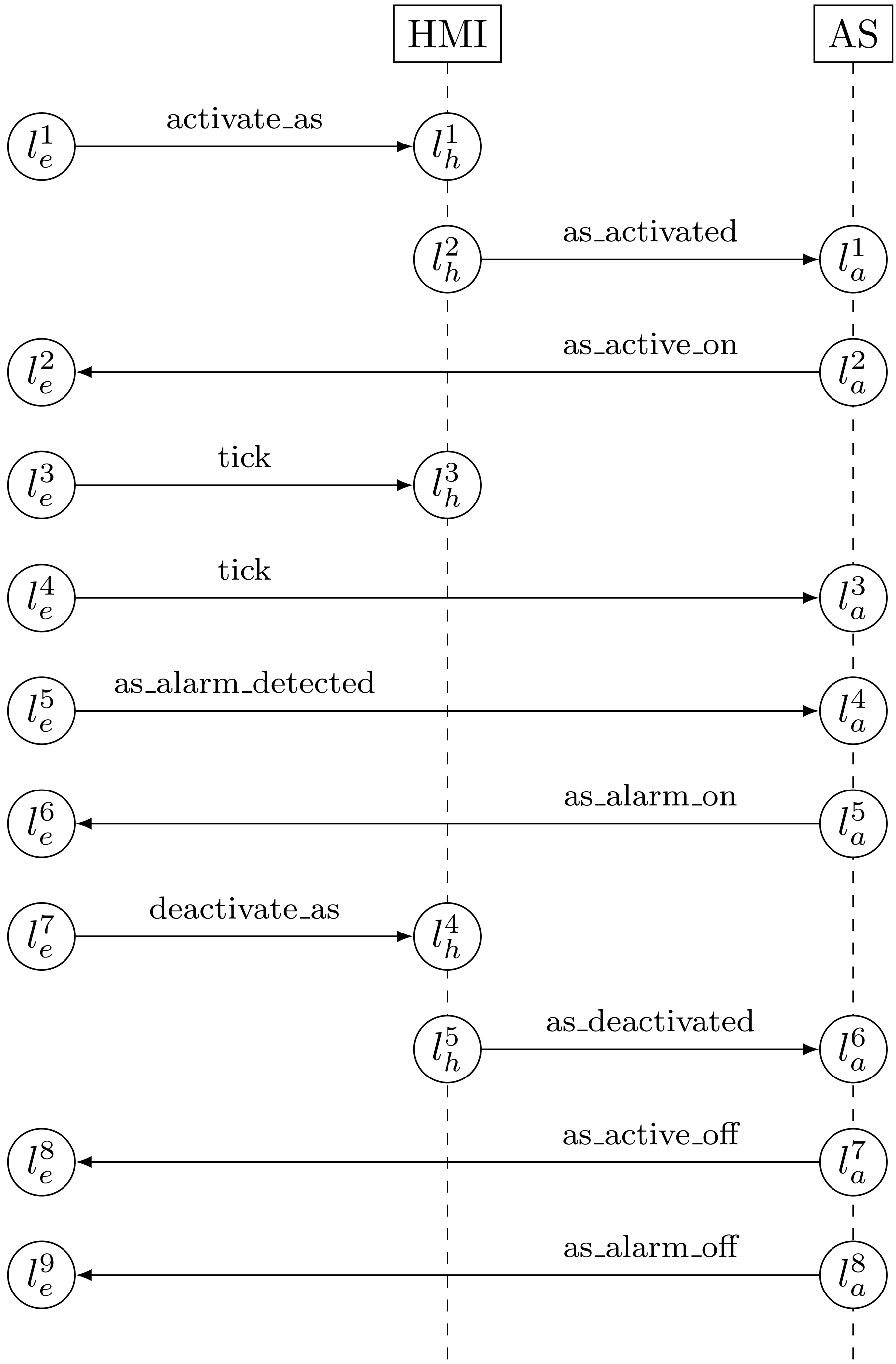

Figure 5.24.: Interaction Test Scenario MSC19

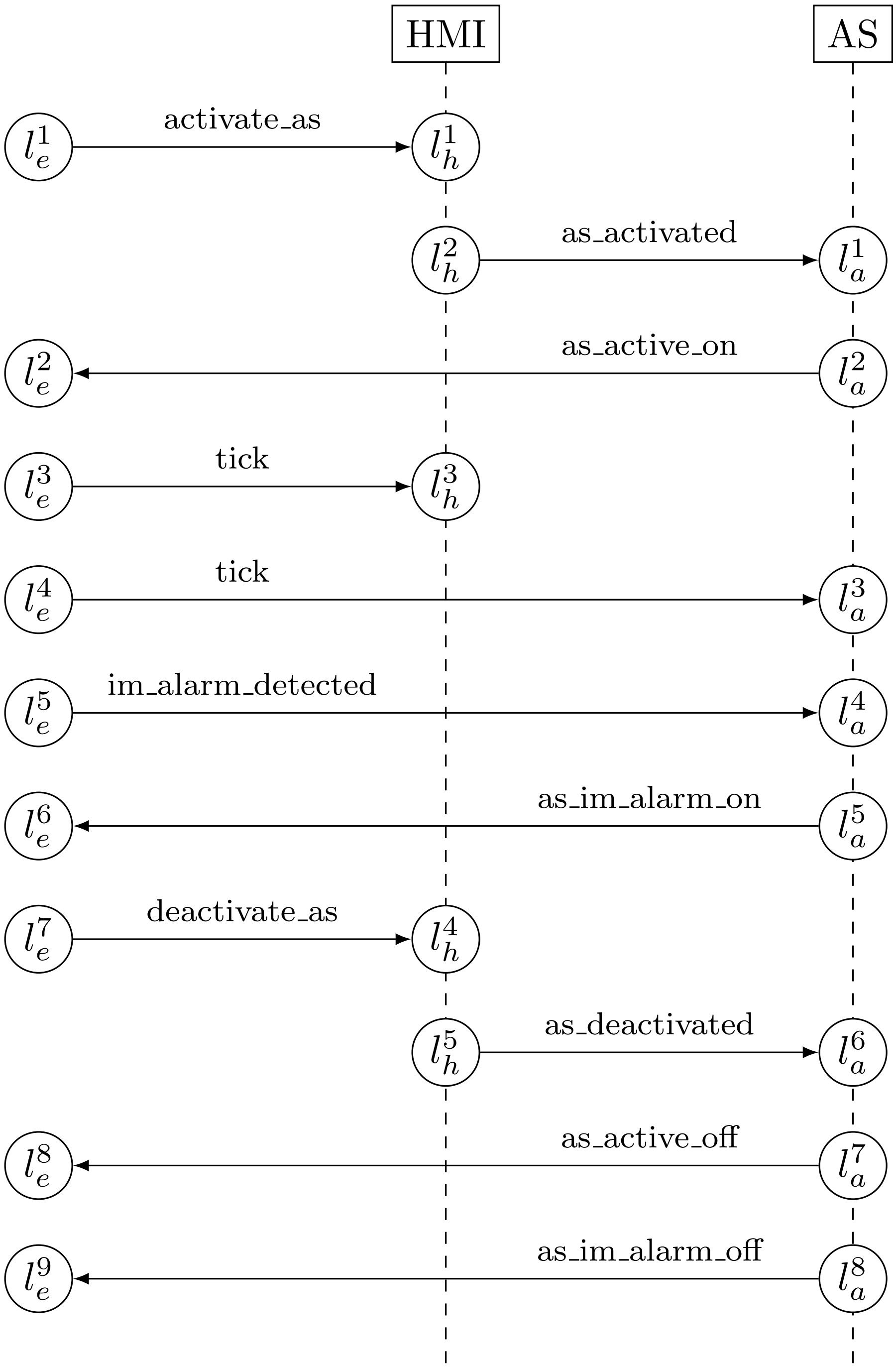

Figure 5.25.: Interaction Test Scenario MSC20

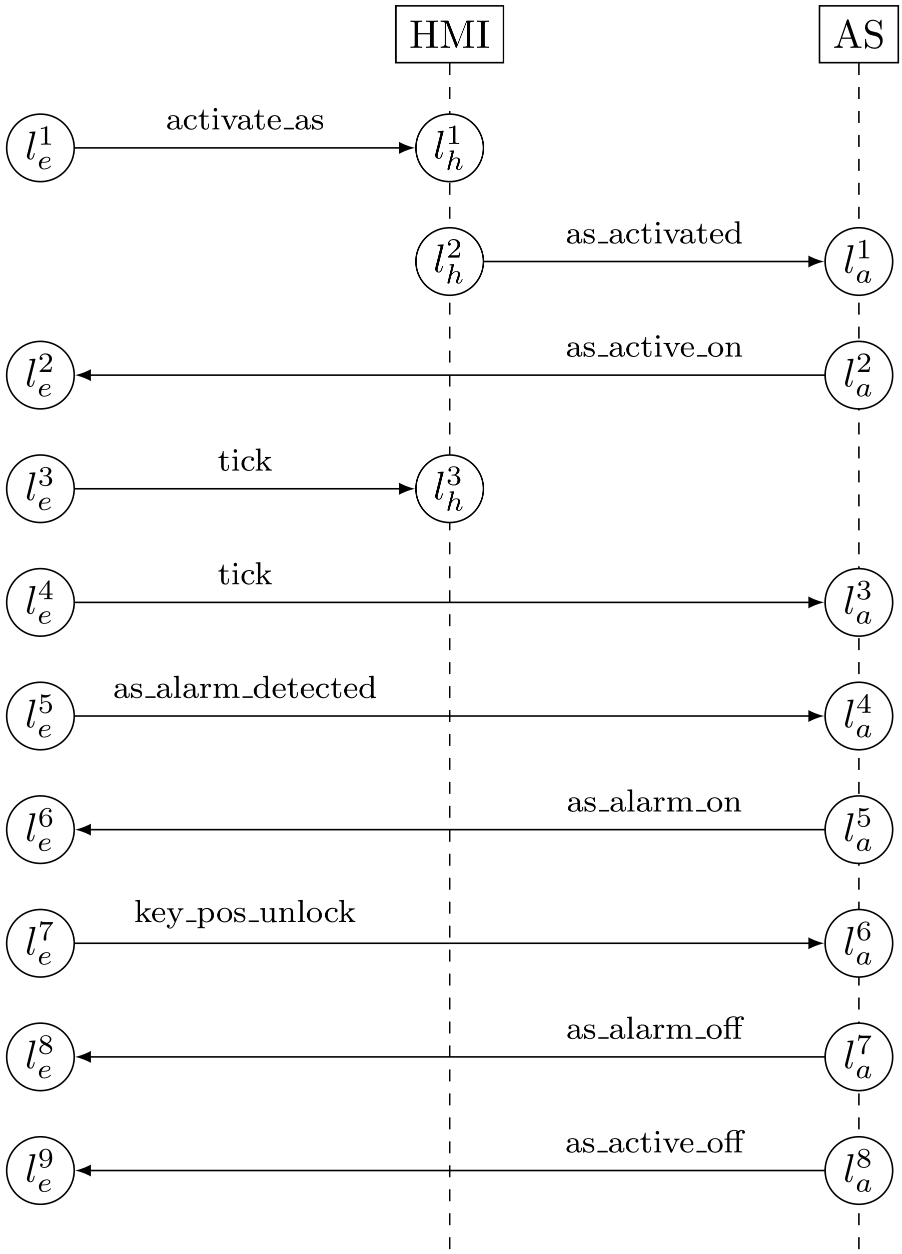

Figure 5.26.: Interaction Test Scenario MSC21

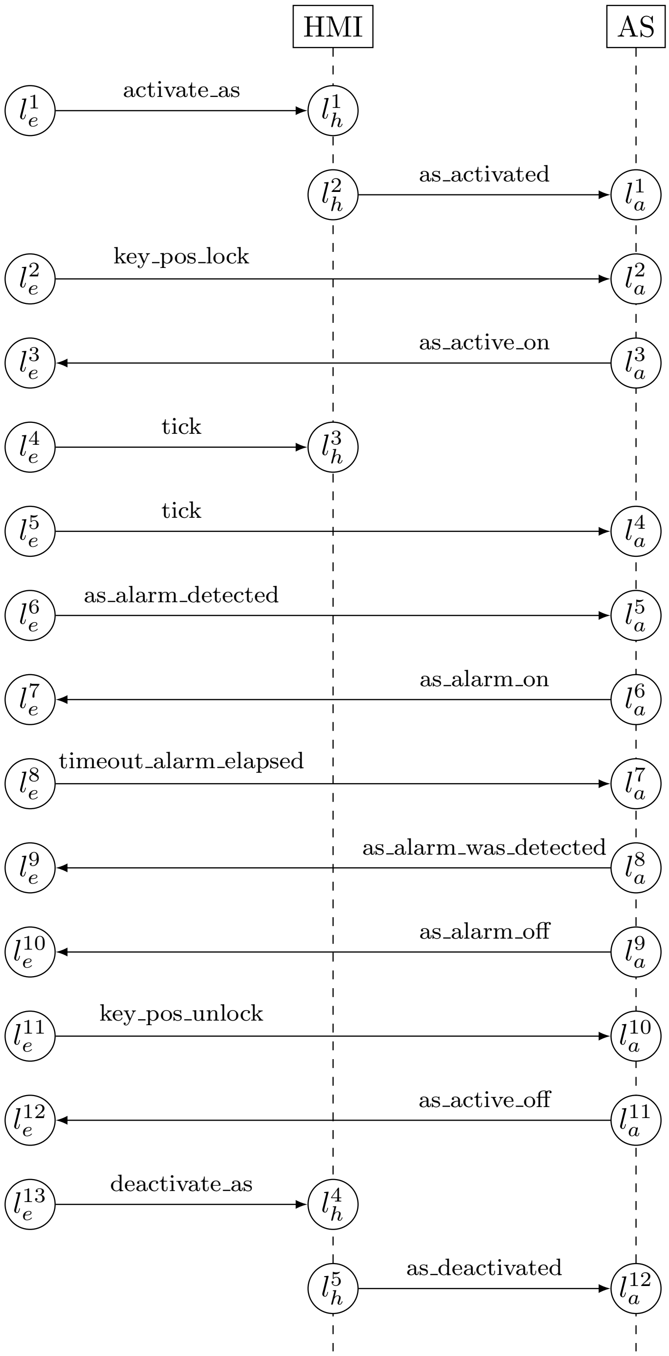

Figure 5.27.: Interaction Test Scenario MSC22

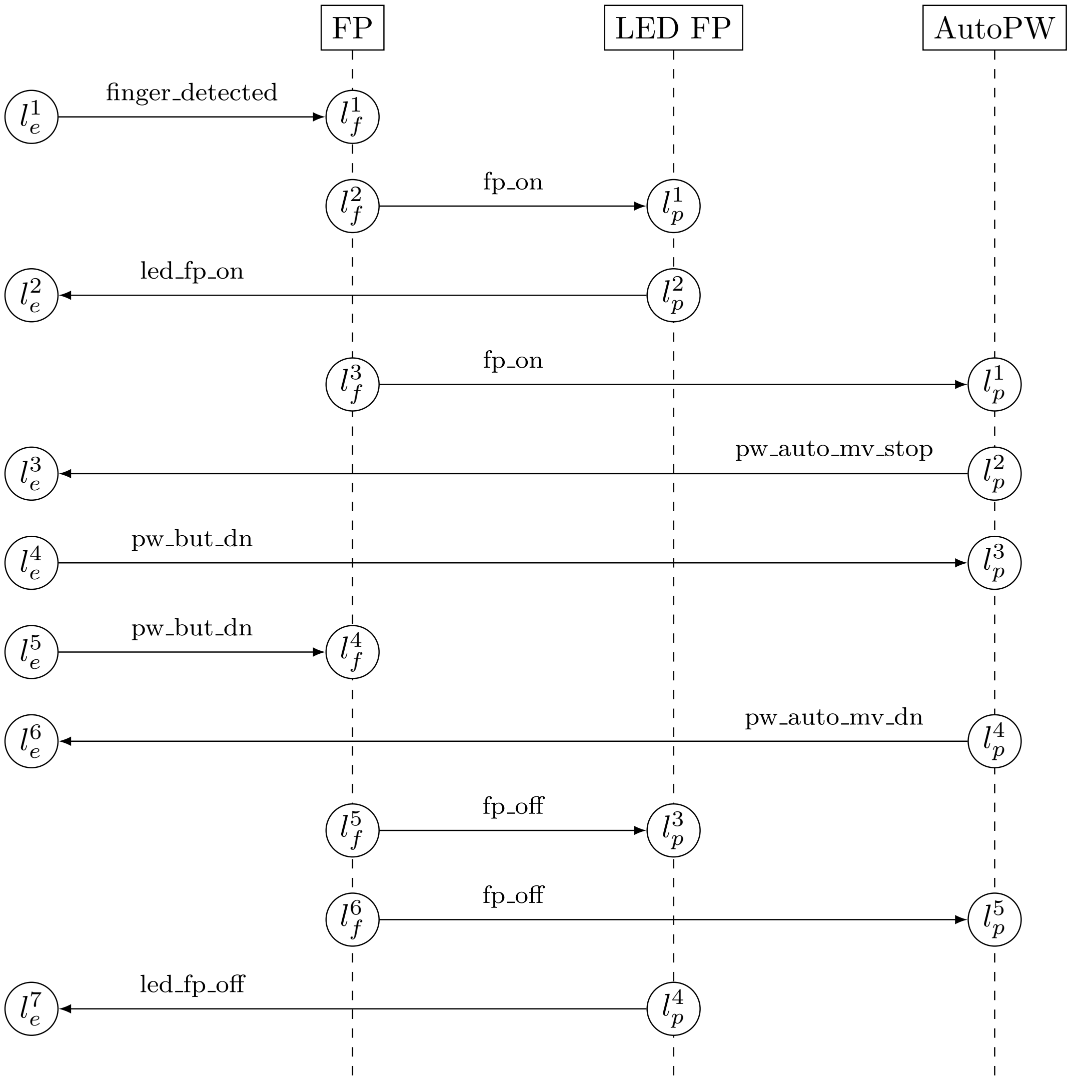

Figure 5.28.: Interaction Test Scenario MSC23

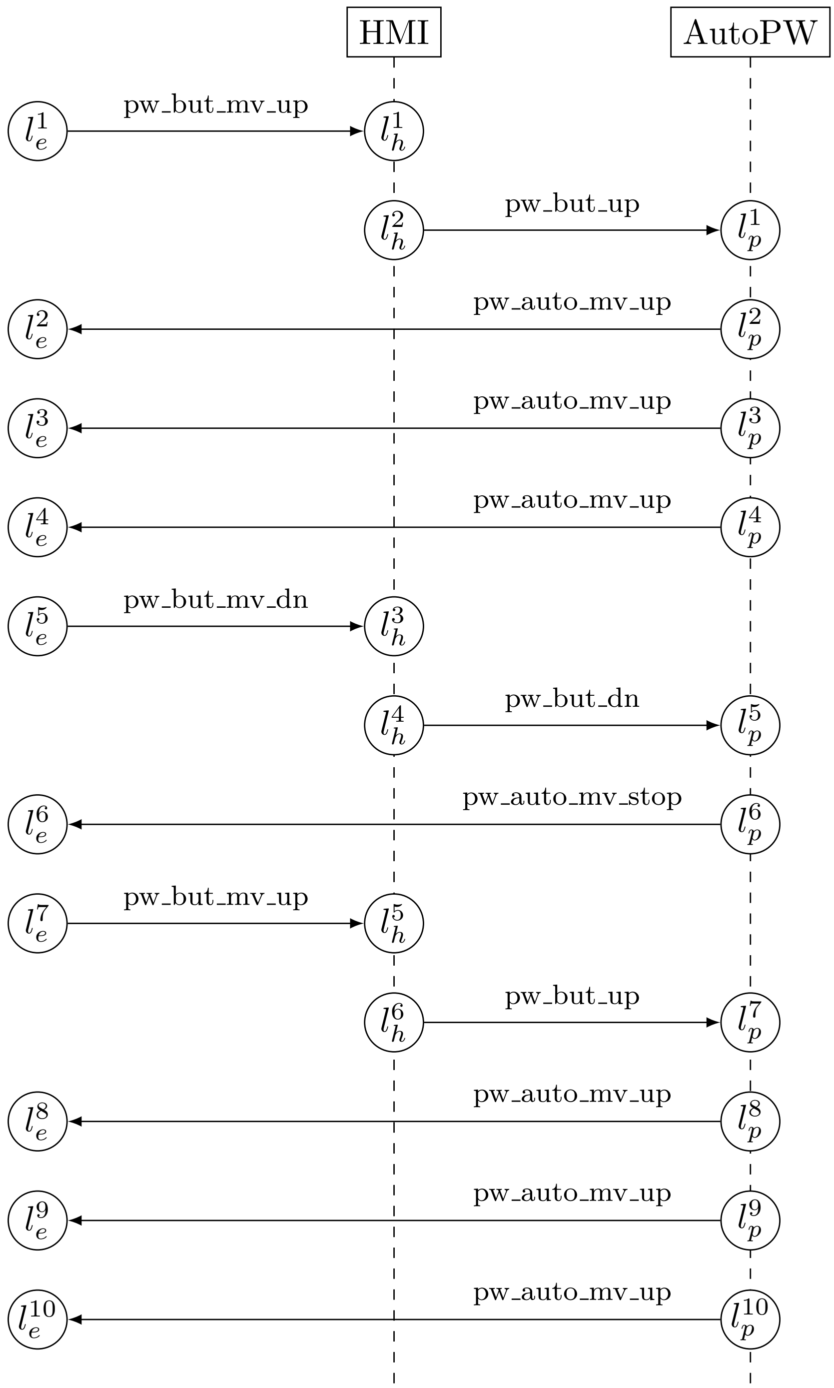

Figure 5.29.: Interaction Test Scenario MSC24

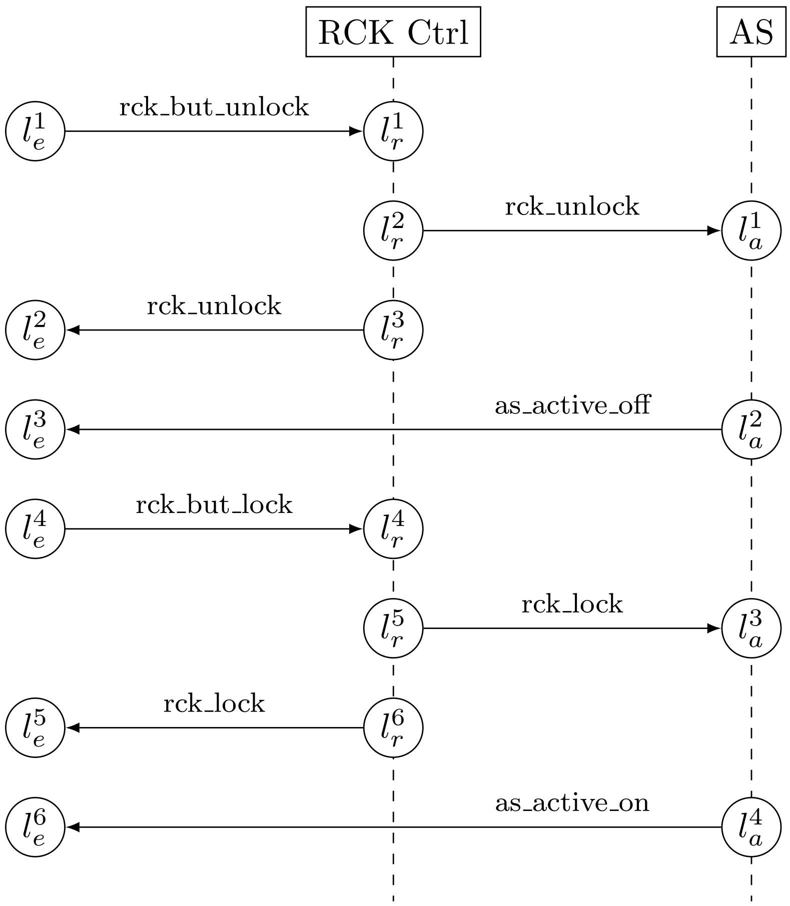

Figure 5.30.: Interaction Test Scenario MSC25

the car using the remote key. In addition, the scenario describes the triggering of the interior alarm and its deactivation, based on the unlocking of the car using the remote key as well as the disabling of the alarm monitoring of the alarm system.

Interaction Test Scenario MSC27 The interaction test scenario *MSC27* shown in Fig. 5.32, describes a scenario between the *Human Machine Interface* (HMI), the *Remote Control Key Controller* (RCK Ctrl) and the *Alarm System* (AS). The scenario defines the unlocking and locking of the car using the remote key and the deactivation and activation of the alarm system via the human machine interface.

Interaction Test Scenario MSC28 The interaction test scenario *MSC28* depicted in Fig. 5.33, defines a scenario between the *Remote Control Key Controller* (RCK Ctrl) and the *Automatic Power Window* (AutoPW). The scenario describes the automated movement of the automatic power window using the remote key. In addition, the scenario defines the blocking of the window movement initiated by the activated finger protection as well as the activated central locking system.

Interaction Test Scenario MSC29 The interaction test scenario *MSC29* defines a scenario between the *Finger Protection* (FP), the *Remote Control Key Controller* (RCK Ctrl) and the *Automatic Power Window* (AutoPW). It is shown in Fig. 5.34 and Fig. 5.35, where the dashed horizontal line represents a cut for better readability. The scenario describes the movement of the automatic power window initiated by the remote key. In addition, the scenario defines the activation of the finger protection, caused by a clamped finger, and the blocking of the automated upwards movement of the power window. Furthermore, the central locking system is active and is blocking the downwards movement of the window.

Interaction Test Scenario MSC30 The interaction test scenario *MSC30* defines a scenario between the *Finger Protection* (FP), the *Remote Control Key Controller* (RCK Ctrl), the *Central Locking System* (CLS) and the *Automatic Power Window* (AutoPW). It is depicted in Fig. 5.36 and Fig. 5.37, where the dashed horizontal line represents a cut for better readability. The scenario describes the unlocking and locking of the central locking system using the remote key. In addition, the scenario defines the movement of the automatic power window initiated by the remote key. The finger protection gets activated by a clamped finger and is further deactivated by pressing the remote button for the downwards movement of the window.

Interaction Test Scenario MSC31 The interaction test scenario *MSC31* shown in Fig. 5.38, defines a scenario between the *Remote Control Key Controller* (RCK Ctrl) and the *Central Locking System* (CLS). The scenario describes the unlocking of the central locking system using the remote key as well as the re-locking, caused by the activation of the safety function.

Interaction Test Scenario MSC32 The interaction test scenario *MSC32* depicted in Fig. 5.39, defines a scenario between the *Remote Control Key Controller* (RCK Ctrl) and the *Alarm System* (AS). The scenario describes the disabling of the alarm monitoring of the alarm system using the remote key and its re-enabling, caused by the activation of the safety function.

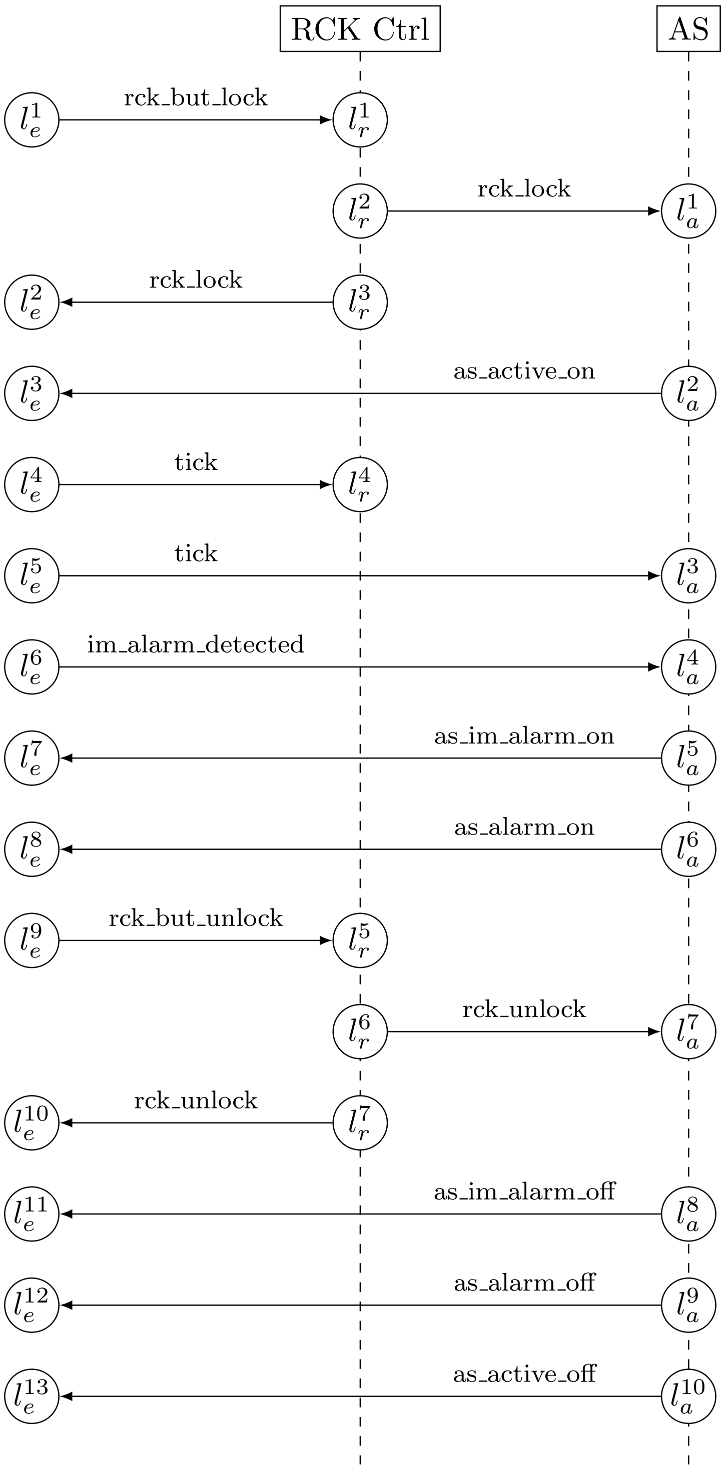

Figure 5.31.: Interaction Test Scenario MSC26

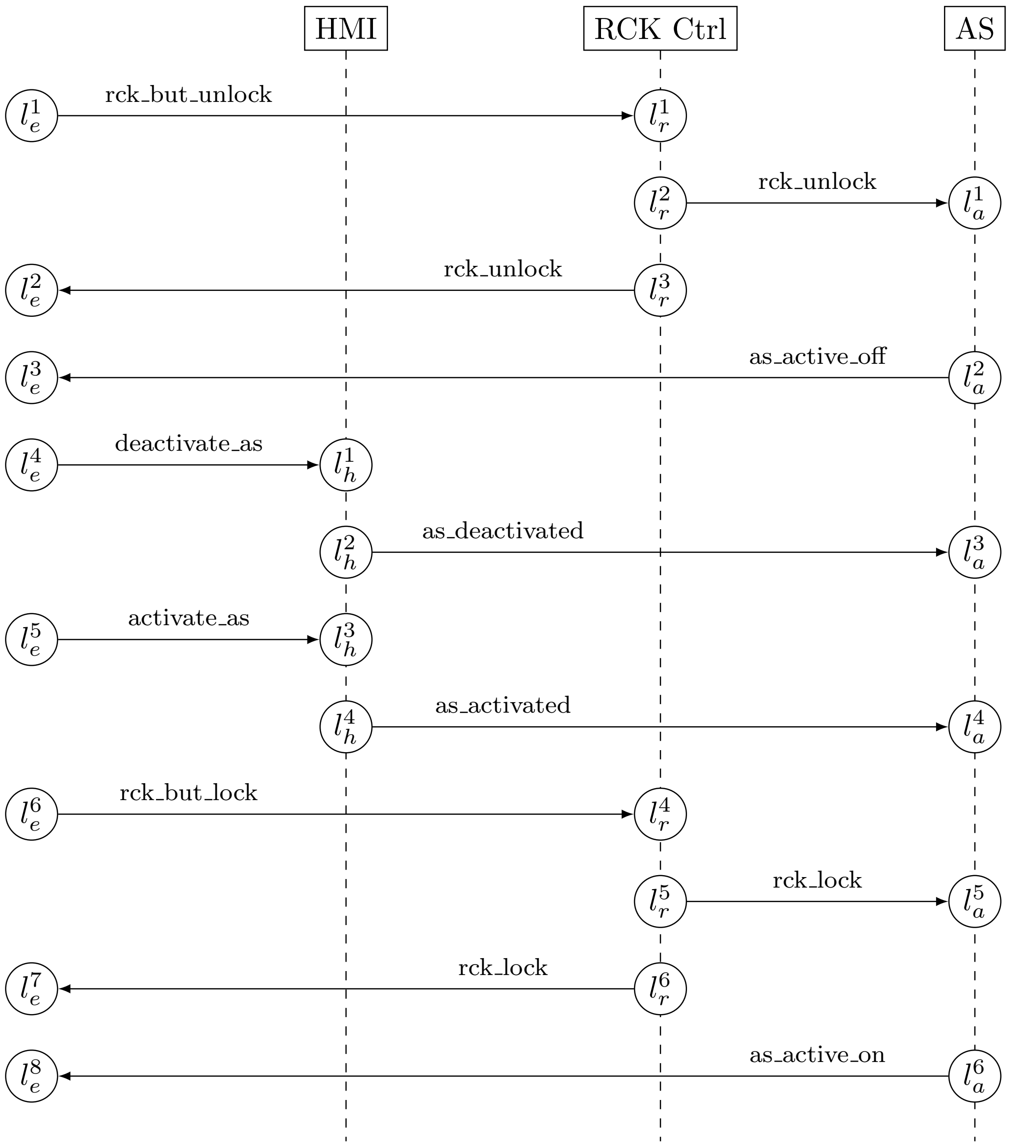

Figure 5.32.: Interaction Test Scenario MSC27

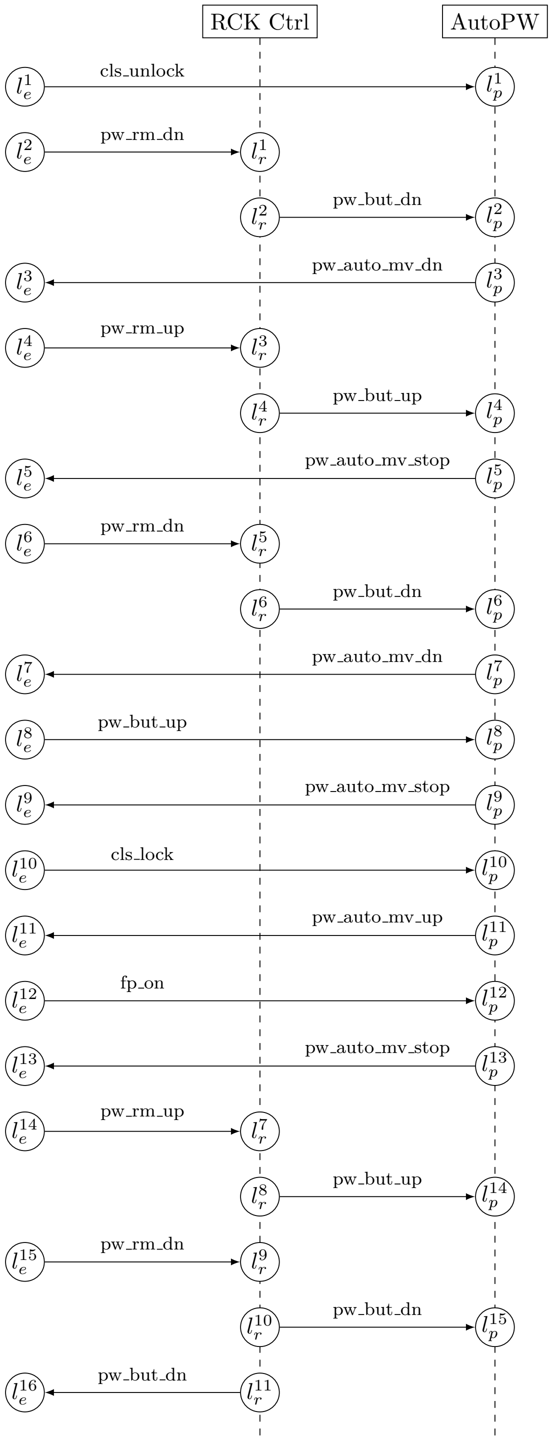

Figure 5.33.: Interaction Test Scenario MSC28

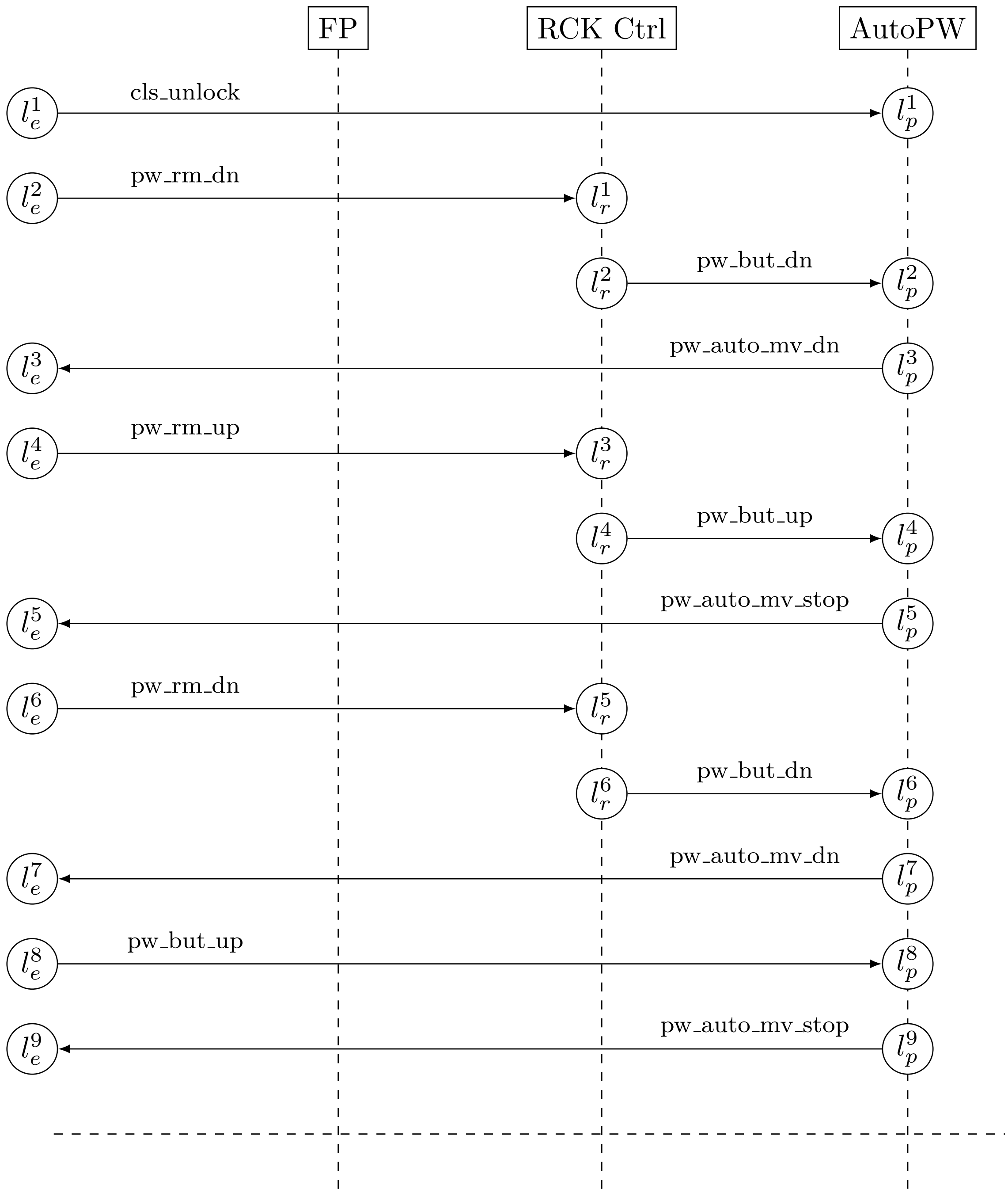

Figure 5.34.: Interaction Test Scenario MSC29 (1)

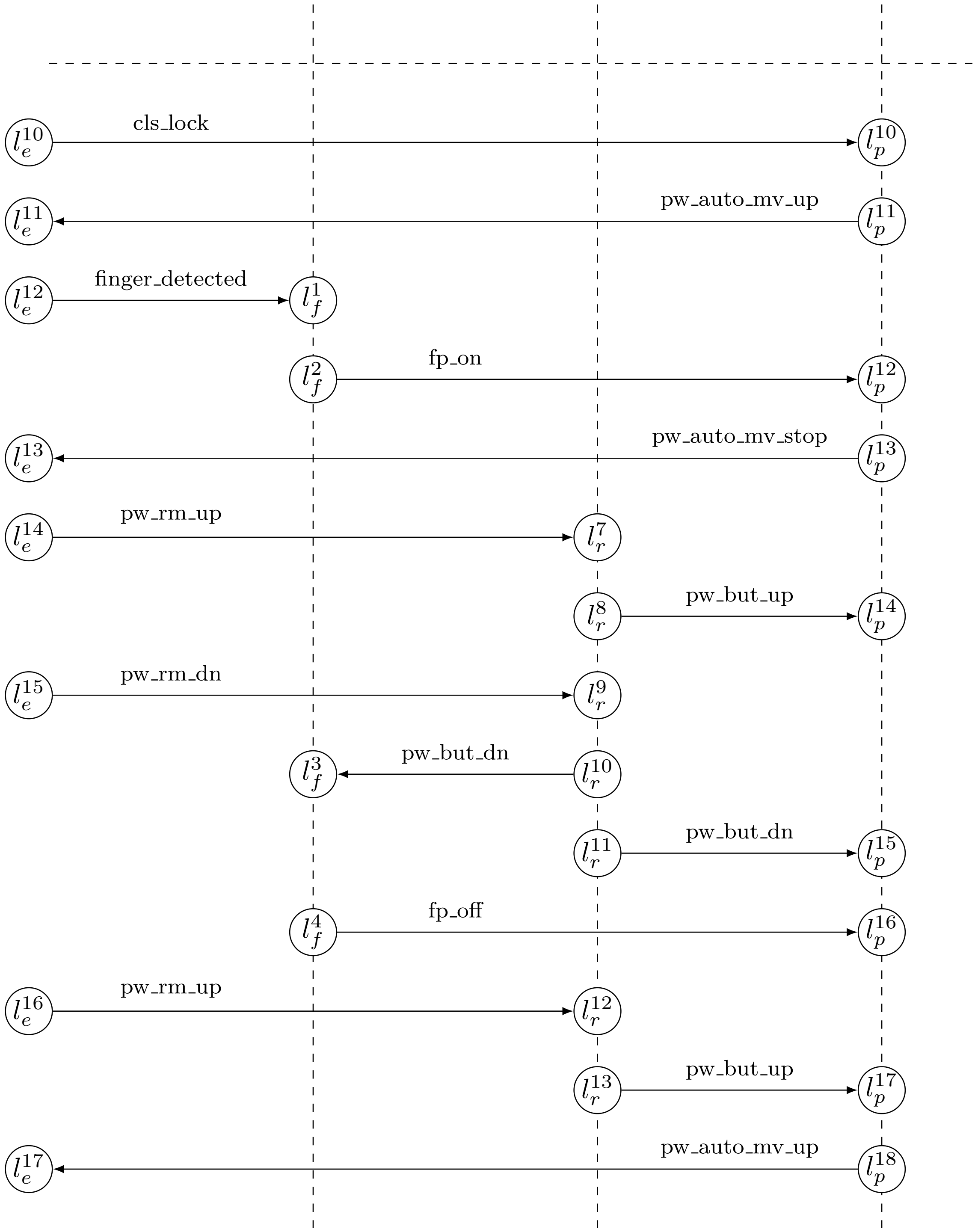

Figure 5.35.: Interaction Test Scenario MSC29 (2)

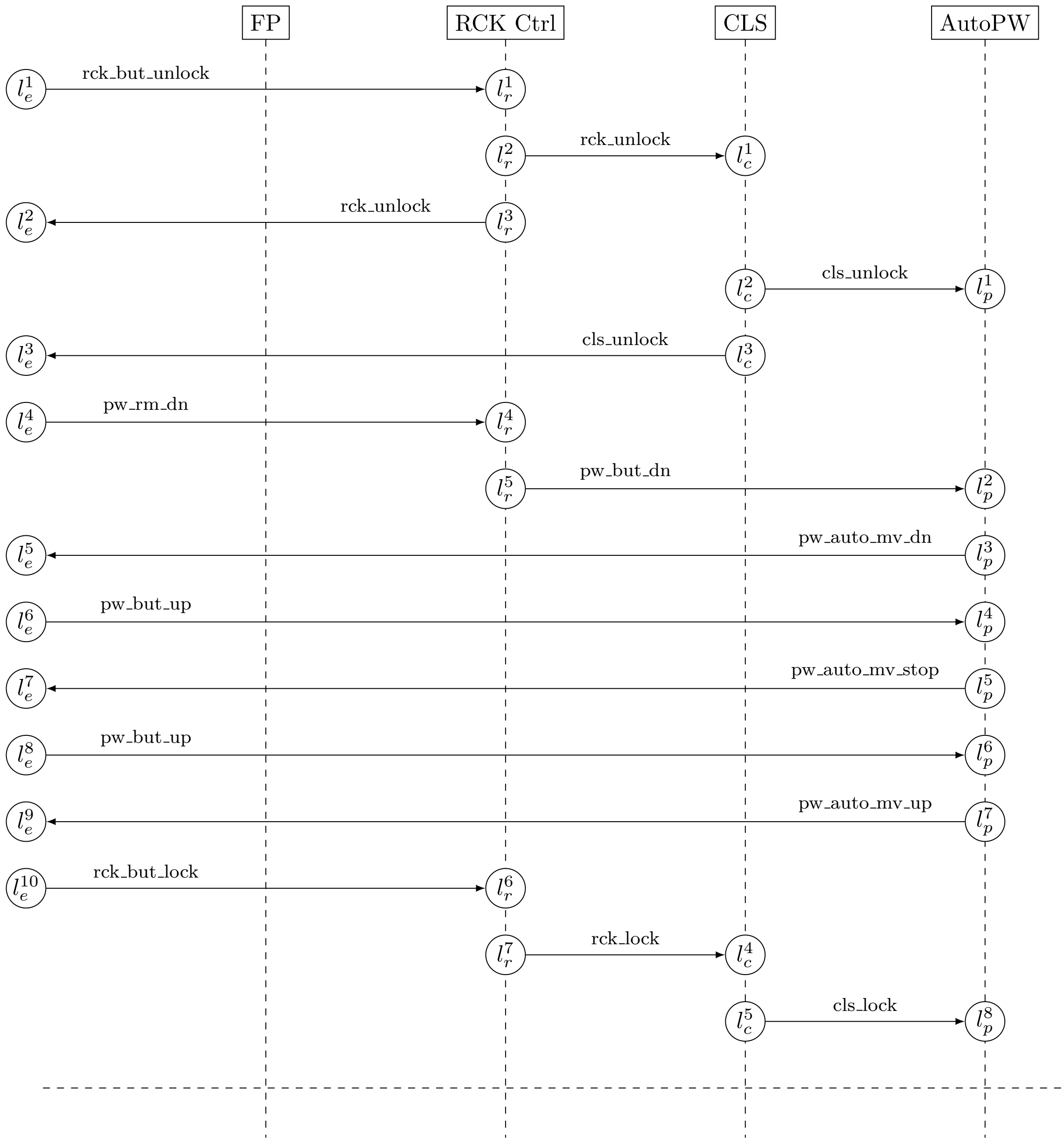

Figure 5.36.: Interaction Test Scenario MSC30 (1)

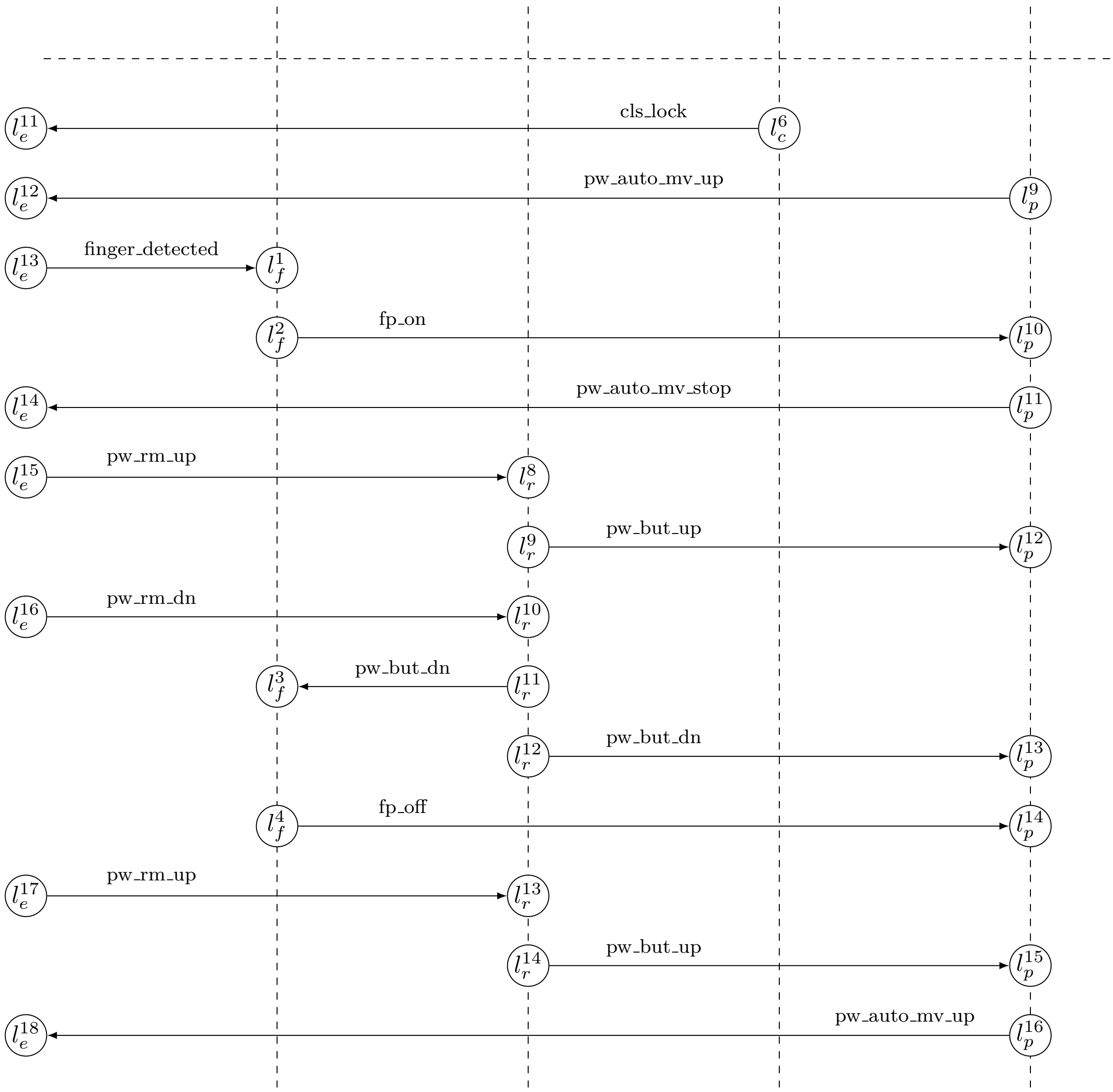

Figure 5.37.: Interaction Test Scenario MSC30 (2)

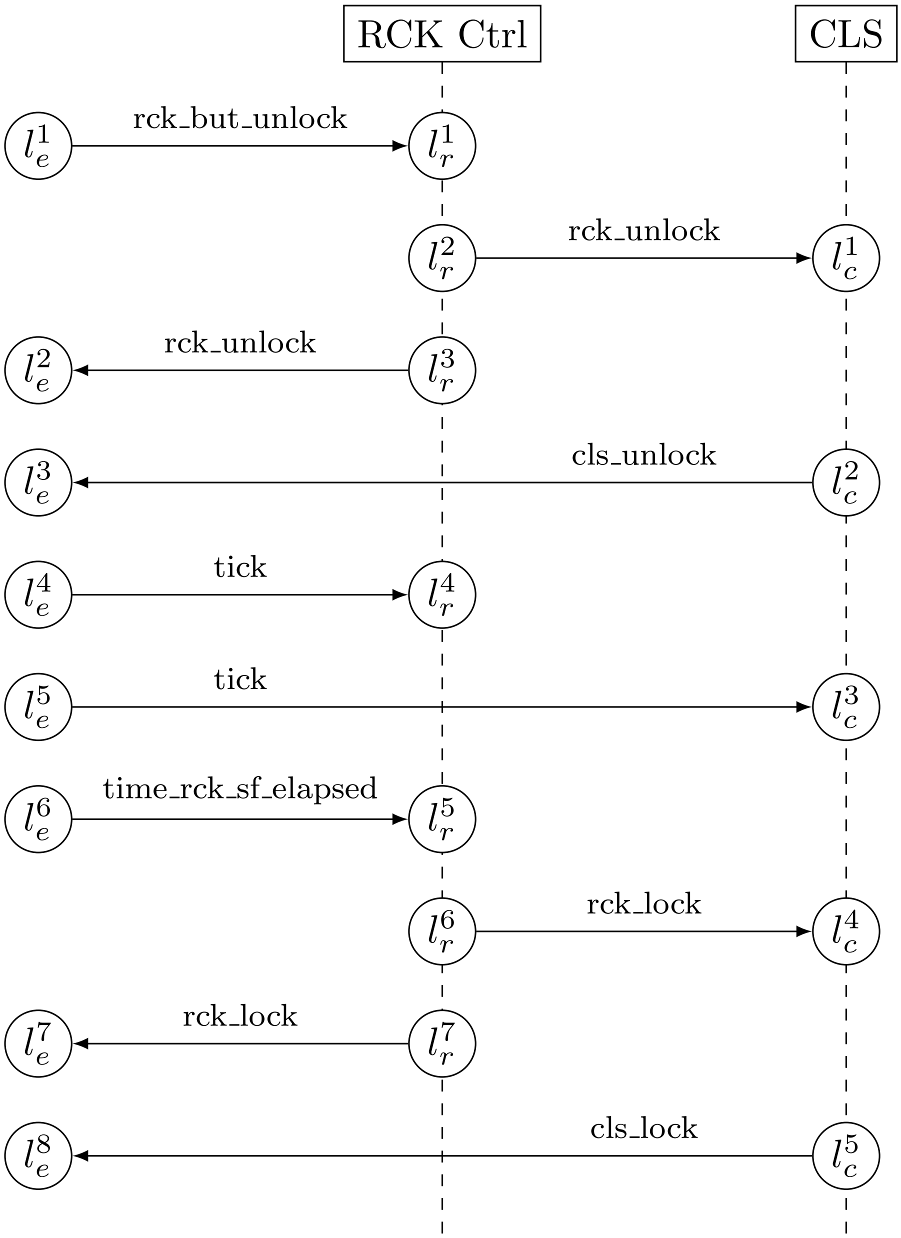

Figure 5.38.: Interaction Test Scenario MSC31

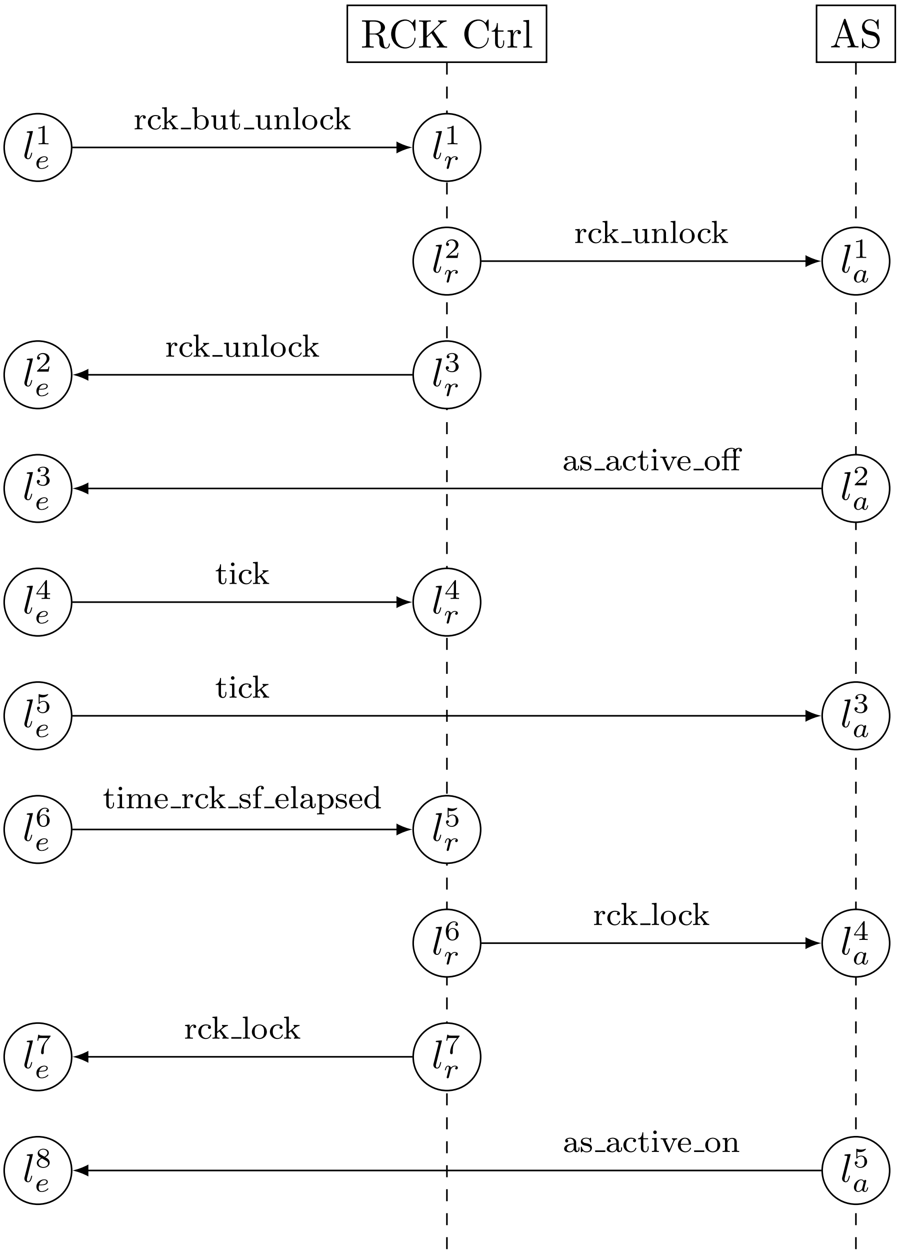

Figure 5.39.: Interaction Test Scenario MSC32

Interaction Test Scenario MSC33 The interaction test scenario *MSC33* shown in Fig. 5.40, defines a scenario between the *Remote Control Key Controller* (RCK Ctrl), the *Central Locking System* (CLS) and the *Alarm System* (AS). The scenario describes the deactivation of the central locking system using the remote key and the disabling of the alarm monitoring of the alarm system. In addition, the safety function gets activated such that after some time (*ticks*) the central locking system is activated again and the alarm monitoring is re-enabled as well.

Interaction Test Scenario MSC34 The interaction test scenario *MSC34* defines a scenario between the *Remote Control Key Controller* (RCK Ctrl) and the *Central Locking System* (CLS). It is depicted in Fig. 5.41 and Fig. 5.42, where the dashed horizontal line represents a cut for better readability. The scenario describes the unlocking and locking of the central locking system using the remote key. In addition, the scenario defines the activation of the safety function as well as the opening of the car such that the safety function is not activated.

Interaction Test Scenario MSC35 The interaction test scenario *MSC35* defines a scenario between the *Human Machine Interface* (HMI), *Exterior Mirror with Heating* (EMH), the *LED Exterior Mirror Bottom* (LED EMB), the *LED Exterior Mirror Top* (LED EMT), the *LED Exterior Mirror Right* (LED EMR), the *LED Exterior Mirror Left* (LED EML) and the *LED Exterior Mirror Heating* (LED EMH). It is shown in Fig. 5.43, Fig. 5.44 and Fig. 5.45, where the dashed horizontal lines represents a cut for better readability. The scenario describes the movement of the exterior mirror as well as the turning on/off of the corresponding LEDs, based on the reaching of the mirror end positions. In addition, the scenario defines the activation and deactivation of the mirror heater.

Interaction Test Scenario MSC36 The interaction test scenario *MSC36* defines a scenario between the *Human Machine Interface* (HMI), the *Finger Protection* (FP), the*Manual Power Window* (ManPW) and the *LED Manual Power Window* (LED ManPW). It is depicted in Fig. 5.46 and Fig. 5.47, where the dashed horizontal line represents a cut for better readability. The scenario describes the upwards and downwards movement of the manual power window and the turning on/off of the corresponding LEDs. In addition, the scenario defines the activation and deactivation of the finger protection and its effects to the window movement.

Interaction Test Scenario MSC37 The interaction test scenario *MSC37* defines a scenario between the *Human Machine Interface* (HMI), the *Alarm System* (AS), the *LED Alarm System Alarm Active* (LED ASAC), the *LED Alarm System Alarm* (LED ASAL) and the *LED Alarm System Alarm Detected* (LED ASAD). It is shown in Fig. 5.48 and Fig. 5.49, where the dashed horizontal line represents a cut for better readability. The scenario describes the activation of the alarm system and the enabling of the alarm monitoring as well as the turning on/off of the corresponding LEDs. In addition, the alarm is triggered and the alarm time elapsed such that the silent alarm is triggered as well. Furthermore, the alarm monitoring of the alarm system is deactivated by unlocking the car and the silent alarm is confirmed via the human machine interface.

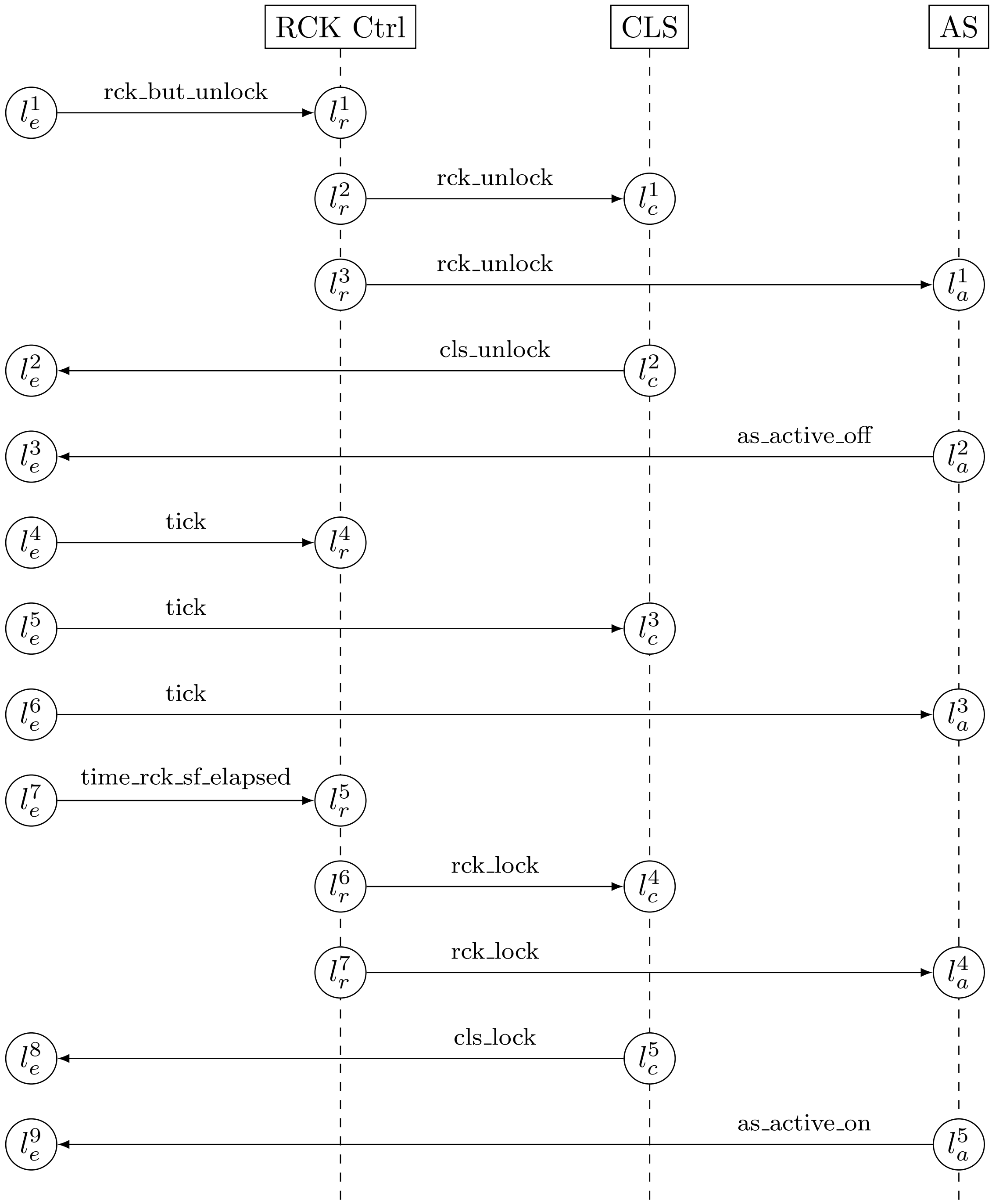

Figure 5.40.: Interaction Test Scenario MSC33

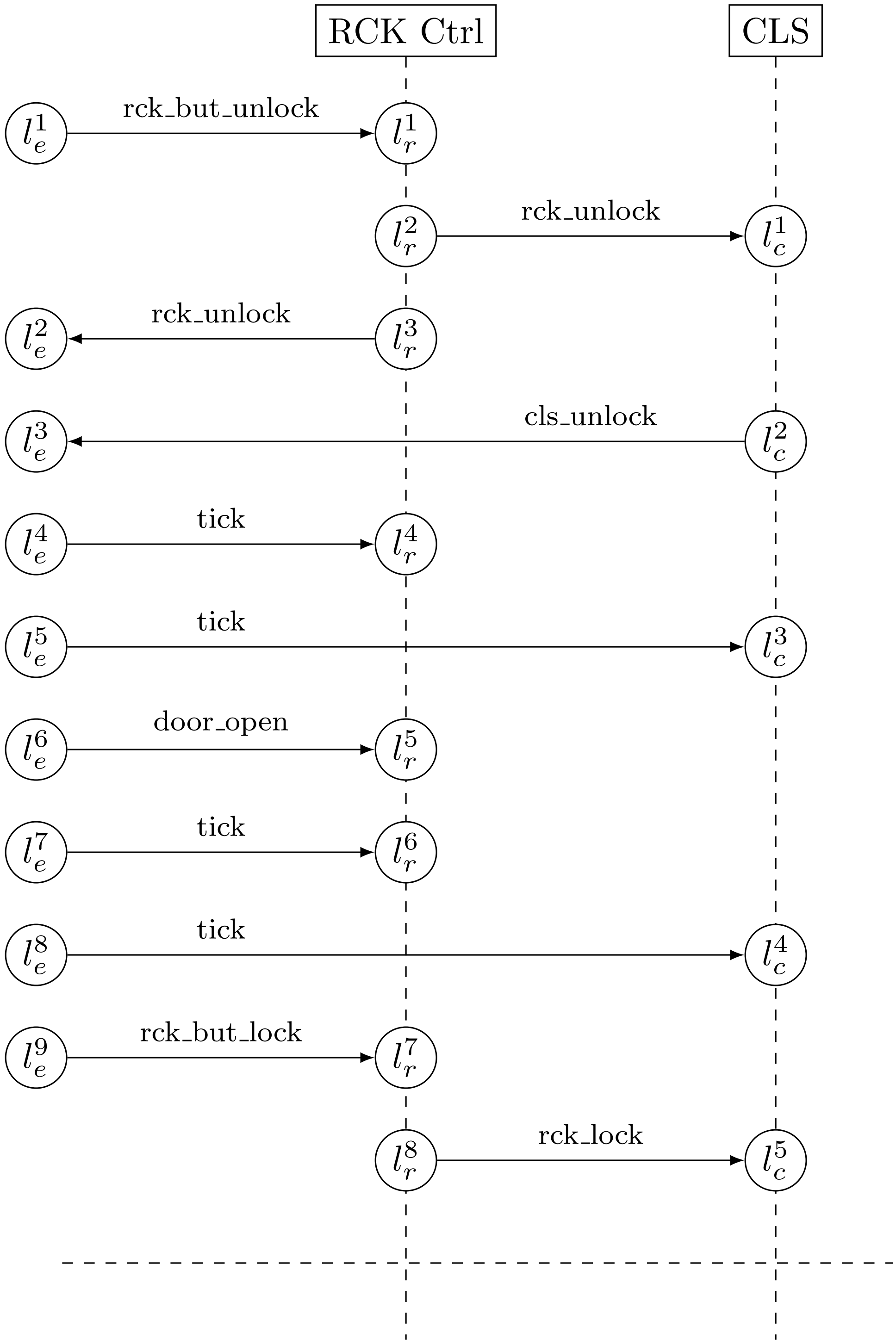

Figure 5.41.: Interaction Test Scenario MSC34 (1)

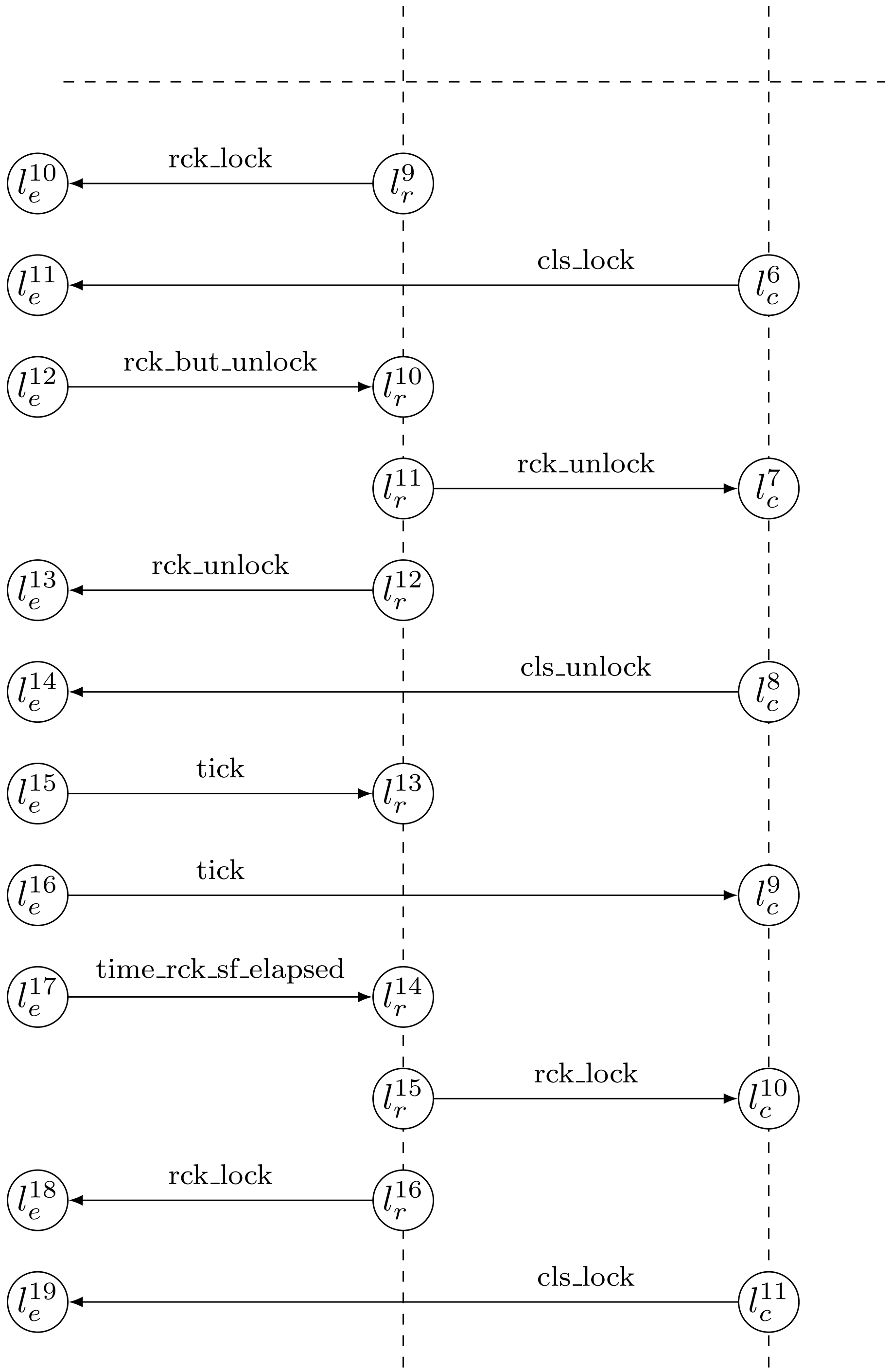

Figure 5.42.: Interaction Test Scenario MSC34 (2)

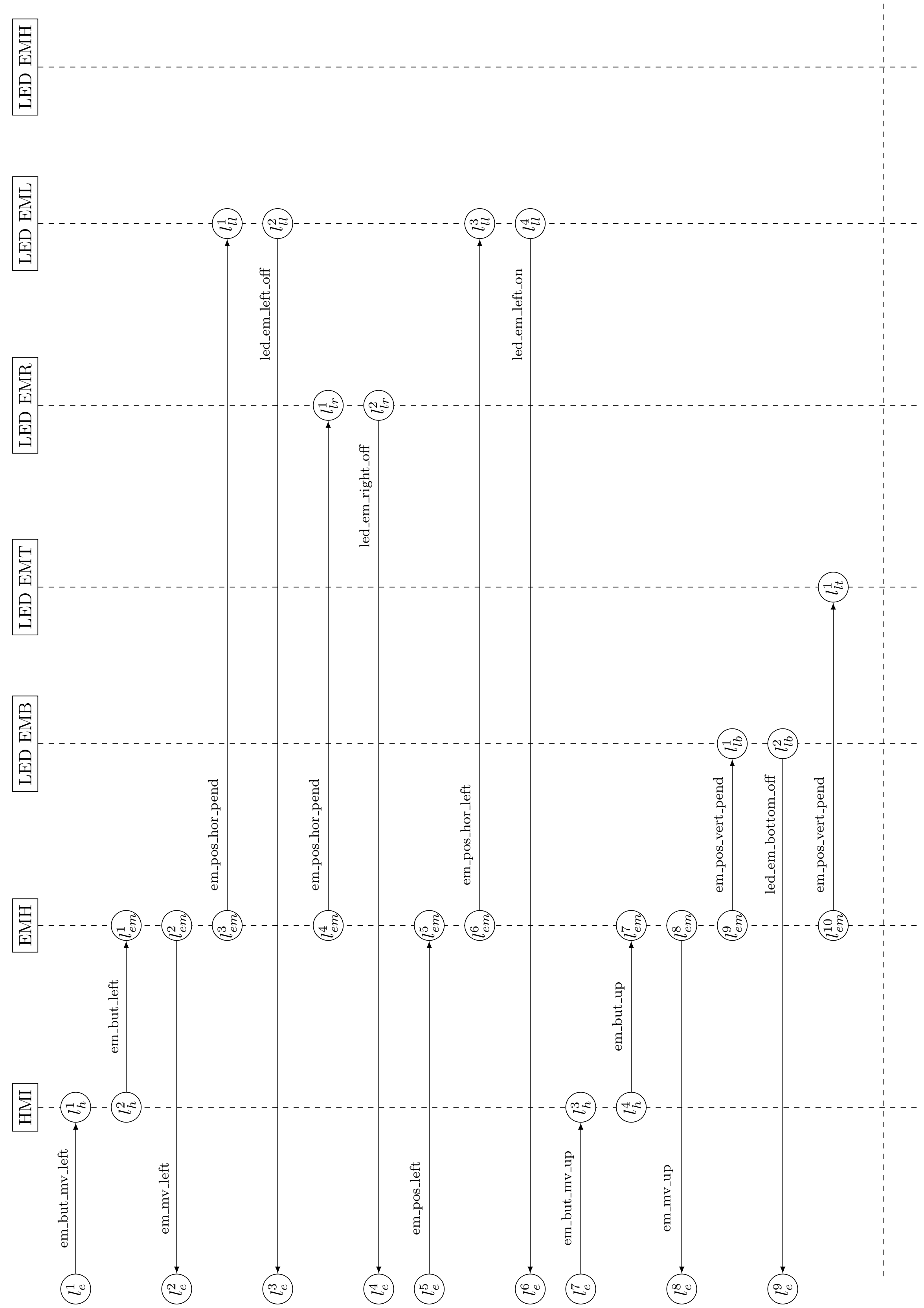

Figure 5.43.: Interaction Test Scenario MSC35 (1)

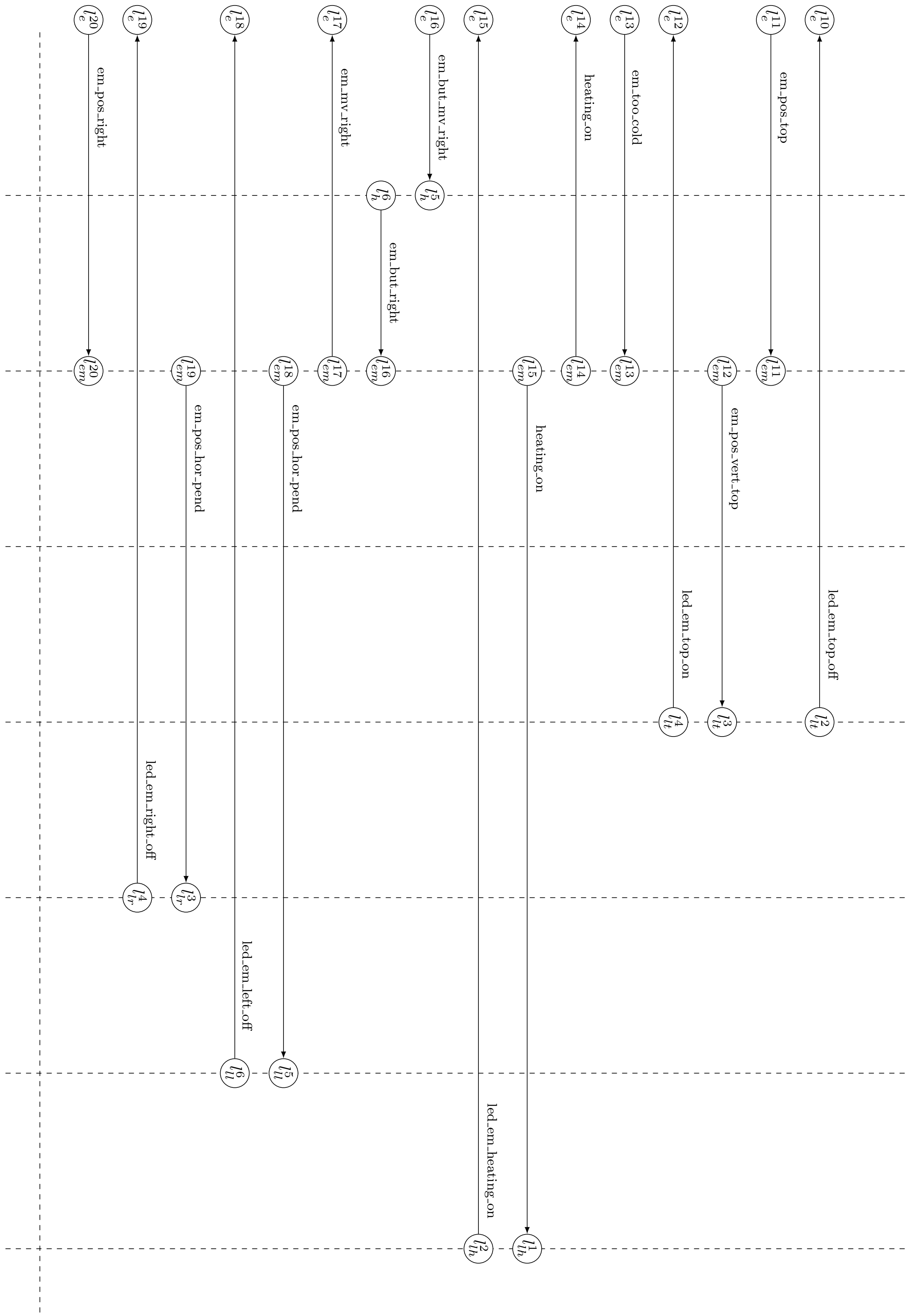

Figure 5.44.: Interaction Test Scenario MSC35 (2)

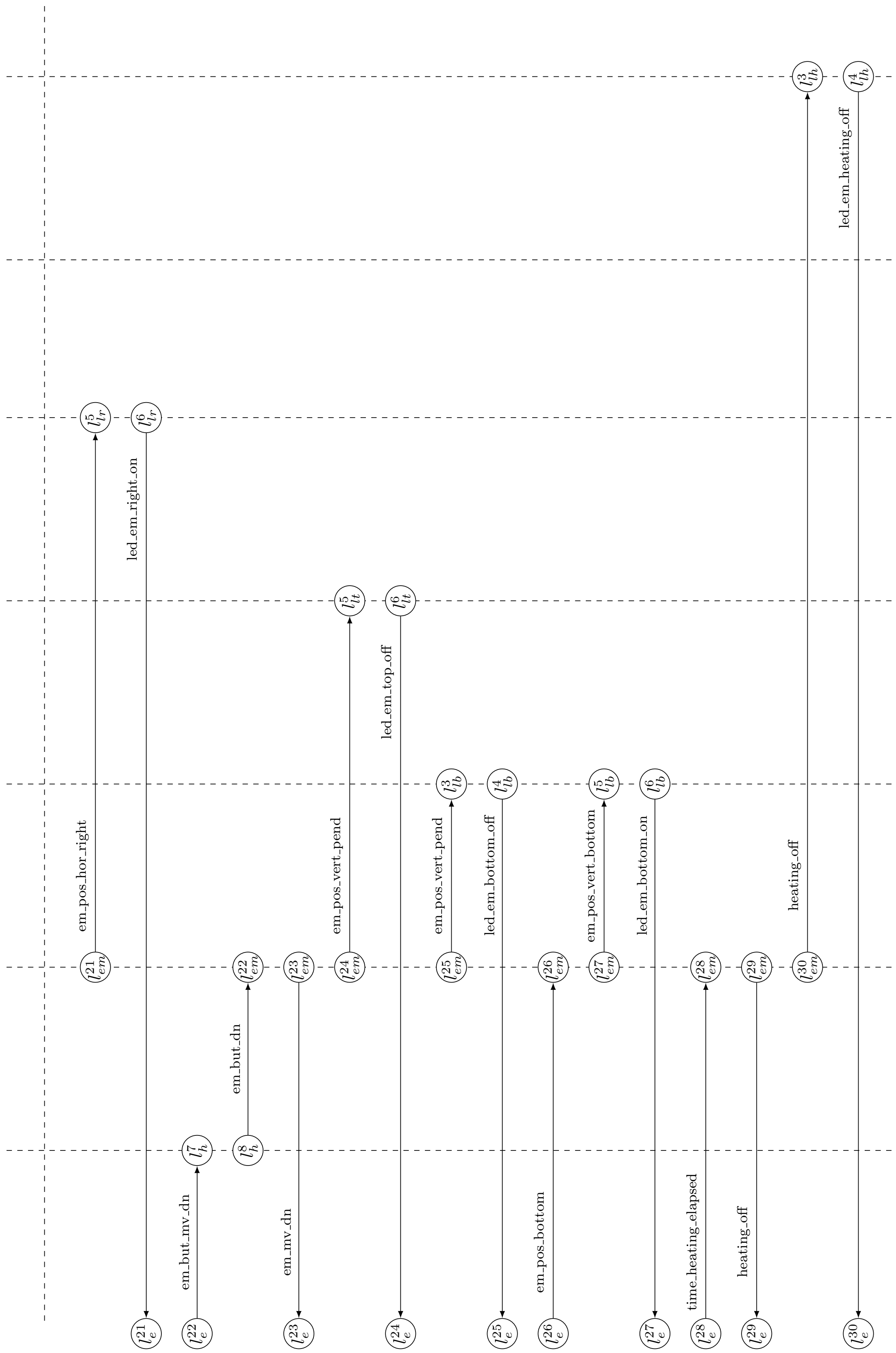

Figure 5.45.: Interaction Test Scenario MSC35 (3)

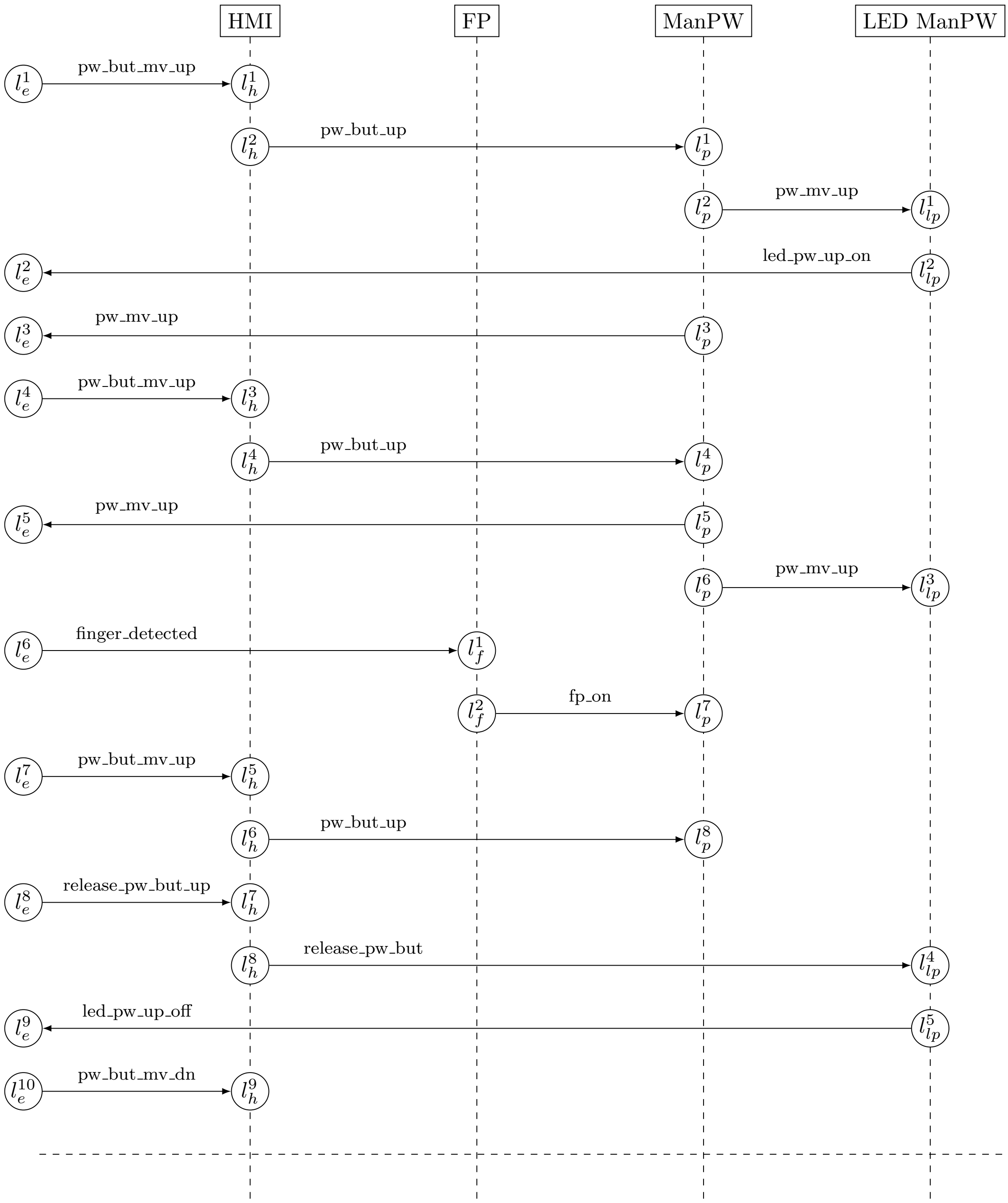

Figure 5.46.: Interaction Test Scenario MSC36 (1)

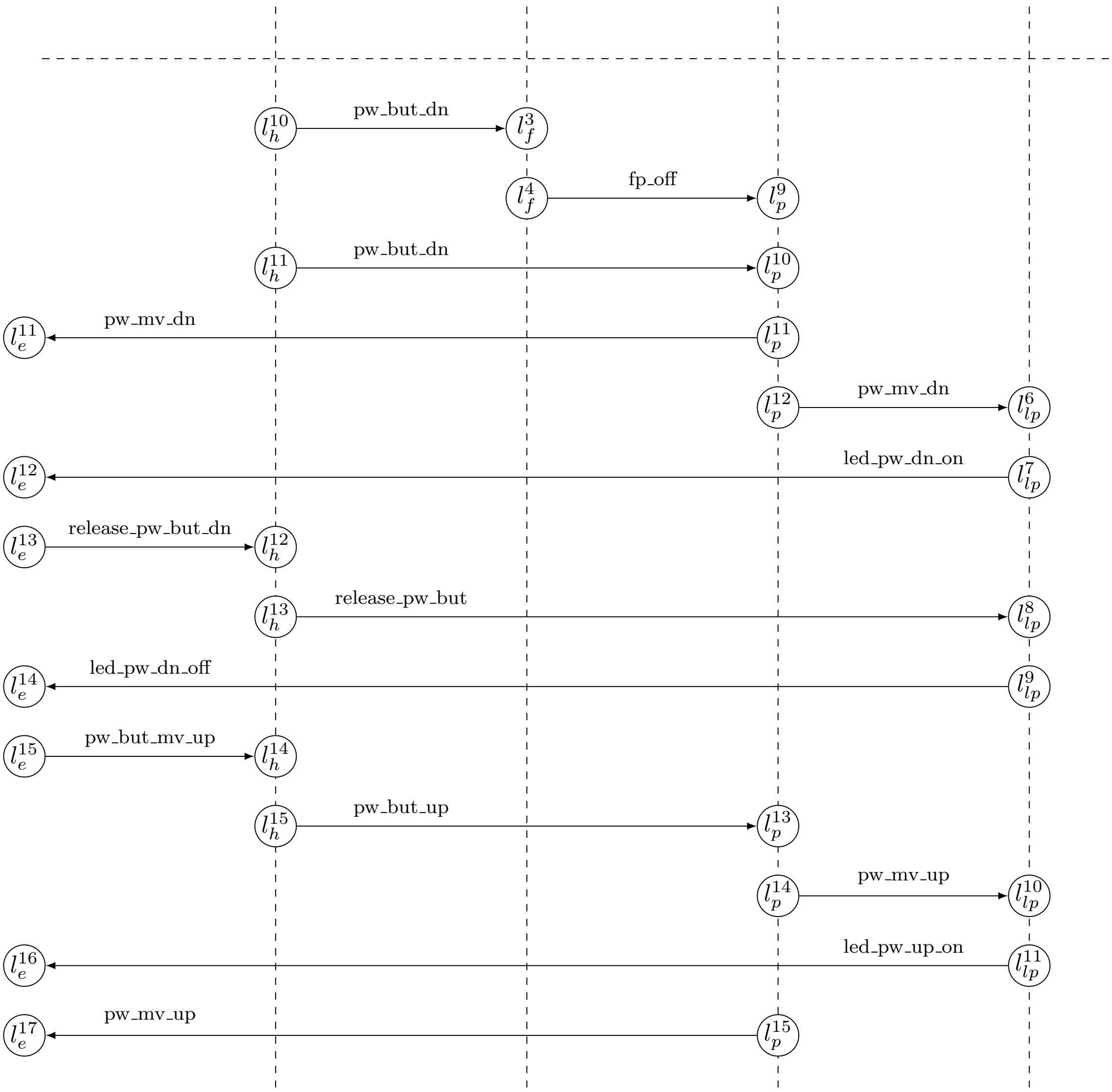

Figure 5.47.: Interaction Test Scenario MSC36 (2)

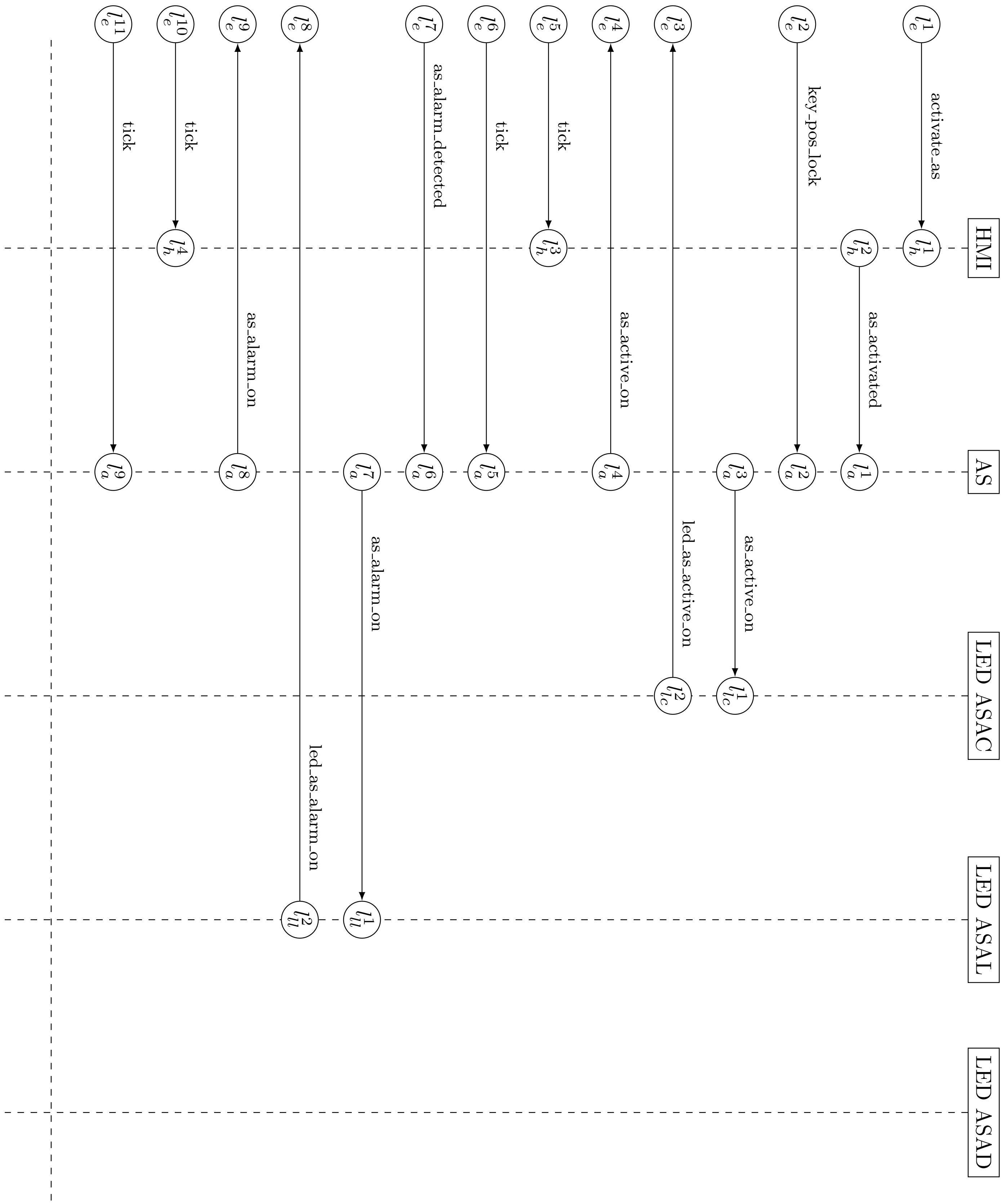

Figure 5.48.: Interaction Test Scenario MSC37 (1)

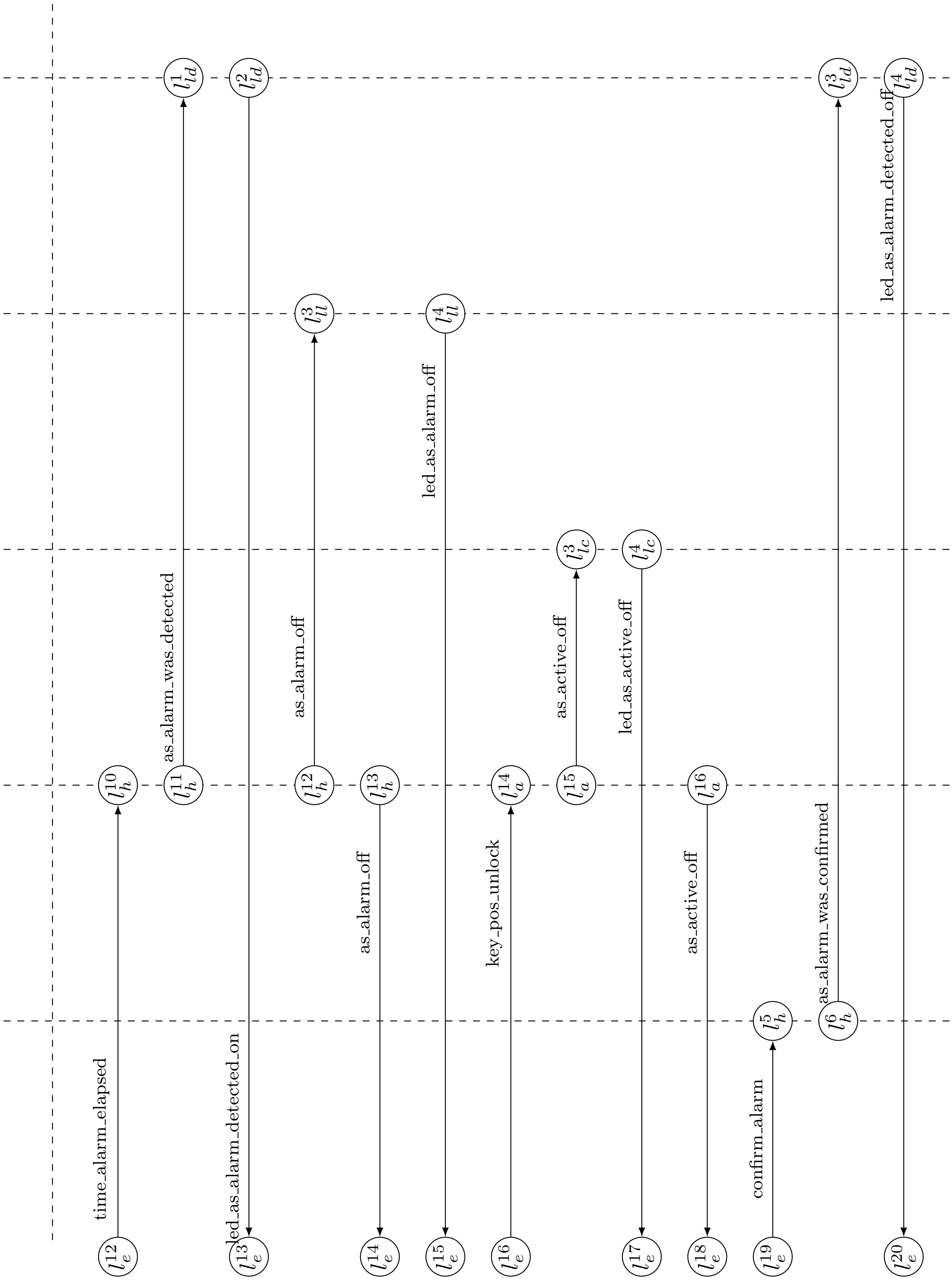

Figure 5.49.: Interaction Test Scenario MSC37 (2)

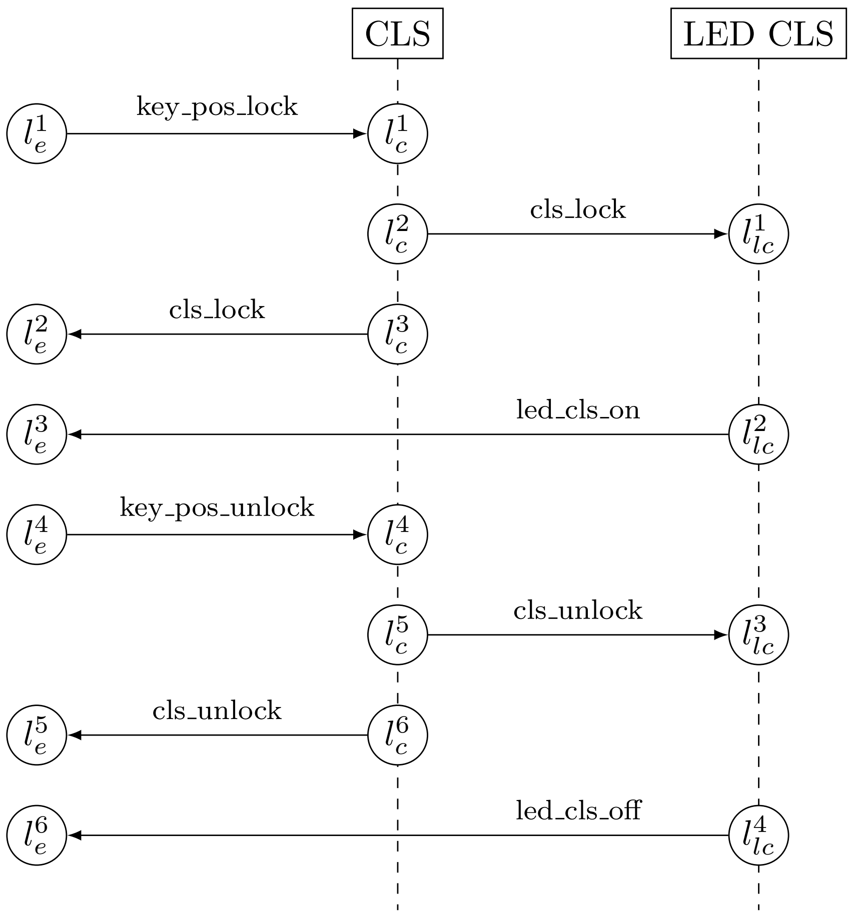

Figure 5.50.: Interaction Test Scenario MSC38

Interaction Test Scenario MSC38 The interaction test scenario *MSC38* depicted in Fig. 5.50, defines a scenario between the *Central Locking System* (CLS) and the *LED Central Locking System* (LED CLS). The scenario describes the turning on/off of the LED for the central locking system, based on the locking and unlocking of the car.

Interaction Test Scenario MSC39 The interaction test scenario *MSC39* shown in Fig. 5.51, defines a scenario between the *Central Locking System* (CLS), the *Manual Power Window* (ManPW) and the *LED Central Locking System* (LED CLS). The scenario describes the locking/unlocking of the central locking system and the turning on/off of the corresponding LED. In addition, the scenario defines the blocking of the window movement, based on the locked central locking system.

Interaction Test Scenario MSC40 The interaction test scenario *MSC40* depicted in Fig. 5.52, defines a scenario between the *Central Locking System* (CLS) and the *Manual Power Window* (ManPW). The scenario describes the blocking of the window movement, based on the activated central locking system and the re-enabling of the window movement after the

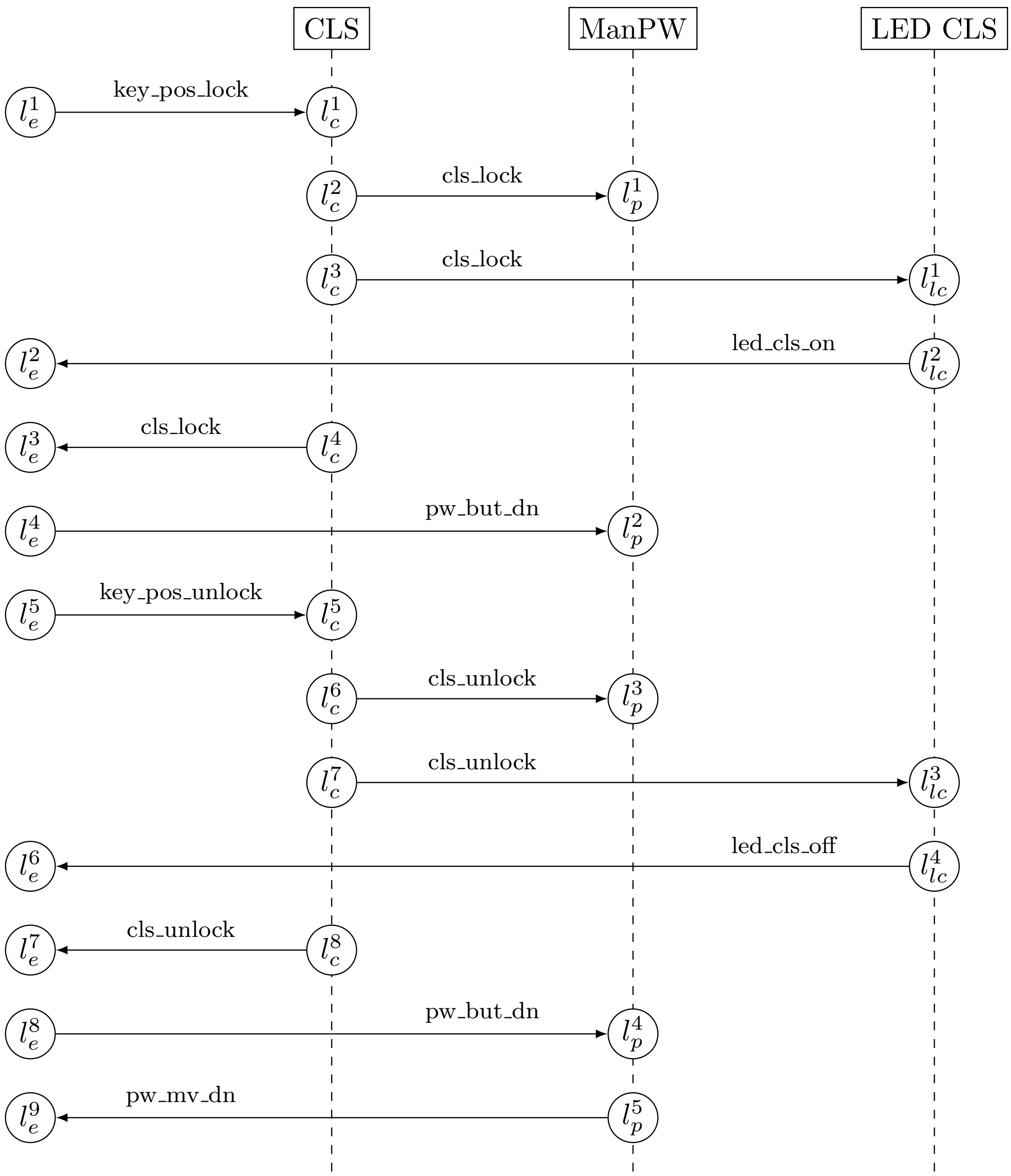

Figure 5.51.: Interaction Test Scenario MSC39

deactivation of the central locking system.

Interaction Test Scenario MSC41 The interaction test scenario *MSC41* defines a scenario between the *Central Locking System* (CLS), the *Finger Protection* (FP) and the *Manual Power Window* (ManPW). It is shown in Fig. 5.53 and Fig. 5.54, where the dashed horizontal line represents a cut for better readability. The scenario describes the downwards movement of the manual power window and its blocking, based on the locking of the central locking system. In addition, the scenario defines the upwards movement of the window while the central locking system is activated and further the activation of the finger protection. The central locking system has first to be deactivated before the window is able to move downwards and the finger protection gets deactivated.

Interaction Test Scenario MSC42 The interaction test scenario *MSC42* depicted in Fig. 5.55, defines a scenario between the *Human Machine Interface* (HMI), the *Central Locking System* (CLS) and the *Manual Power Window* (ManPW). The scenario describes the blocking and release of the downwards movement of the manual power window, based on the locking state of the central locking system.

Interaction Test Scenario MSC43 The interaction test scenario *MSC43* defines a scenario between the *Human Machine Interface* (HMI), the *Central Locking System* (CLS), the *Finger Protection* (FP) and the *Manual Power Window* (ManPW). It is shown in Fig. 5.56 and Fig. 5.57, where the dashed horizontal line represents a cut for better readability. The scenario describes the downwards movement of the manual power window initiated via the human machine interface and its blocking, based on the activation of the central locking system. In addition, the scenario defines the upwards movement of the window via the human machine interface while the central locking system is activated and further the activation of the finger protection. The central locking system has first to be deactivated before the window is able to move downwards and the finger protection gets deactivated.

Interaction Test Scenario MSC44 The interaction test scenario *MSC44* defines a scenario between the *Remote Control Key Controller* (RCK Ctrl), the *Central Locking System* (CLS), the *Finger Protection* (FP) and the *Manual Power Window* (Man PW). It is shown in Fig. 5.58 and Fig. 5.59, where the dashed horizontal line represents a cut for better readability. The scenario describes the downwards movement of the manual power window and its blocking, based on the activation of the central locking system initiated by the remote key. In addition, the scenario defines the upwards movement of the window while the central locking system is activated and further the activation of the finger protection. The central locking system has first to be deactivated via the remote key before the window is able to move downwards and the finger protection gets deactivated.

Interaction Test Scenario MSC45 The interaction test scenario *MSC45* depicted in Fig. 5.60, defines a scenario between the *Remote Control Key Controller* (RCK Ctrl), the *Central Locking System* (CLS) and the *LED Central Locking System* (LED CLS). The scenario describes the activation and deactivation of the central locking system via the remote key and the turning

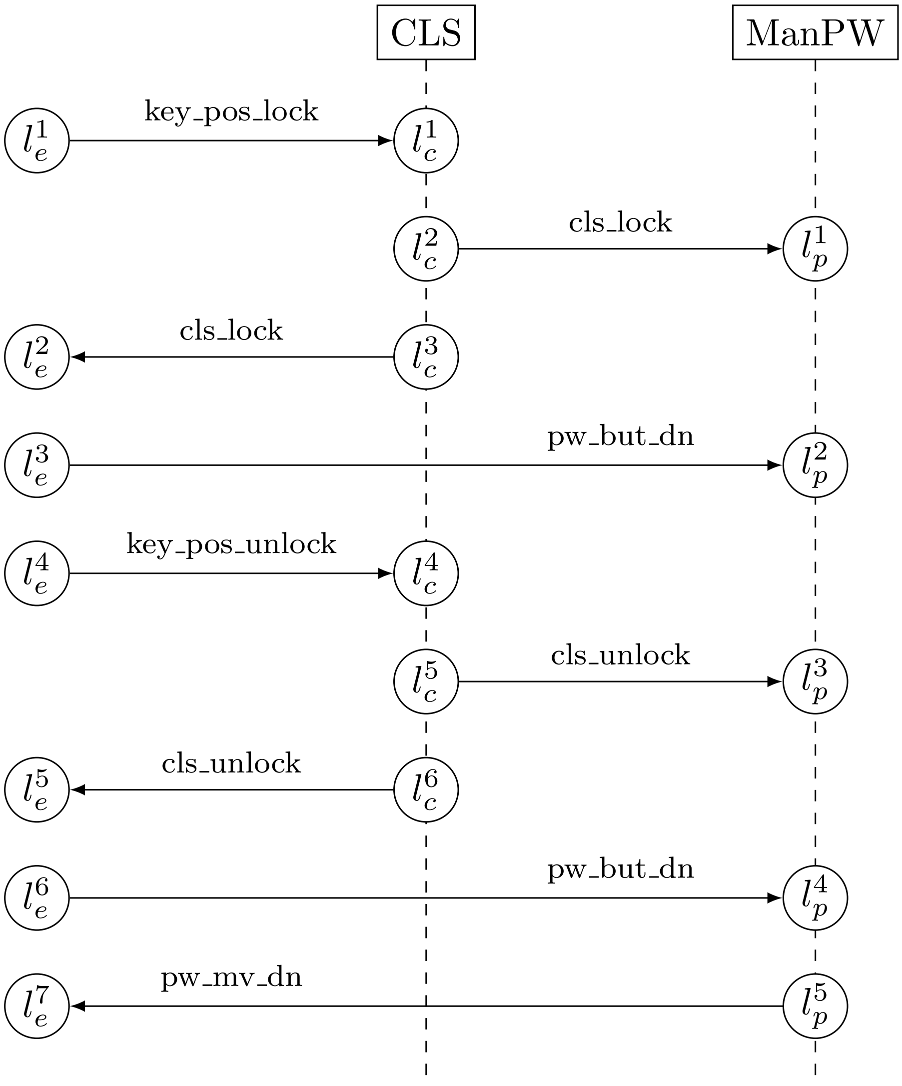

Figure 5.52.: Interaction Test Scenario MSC40

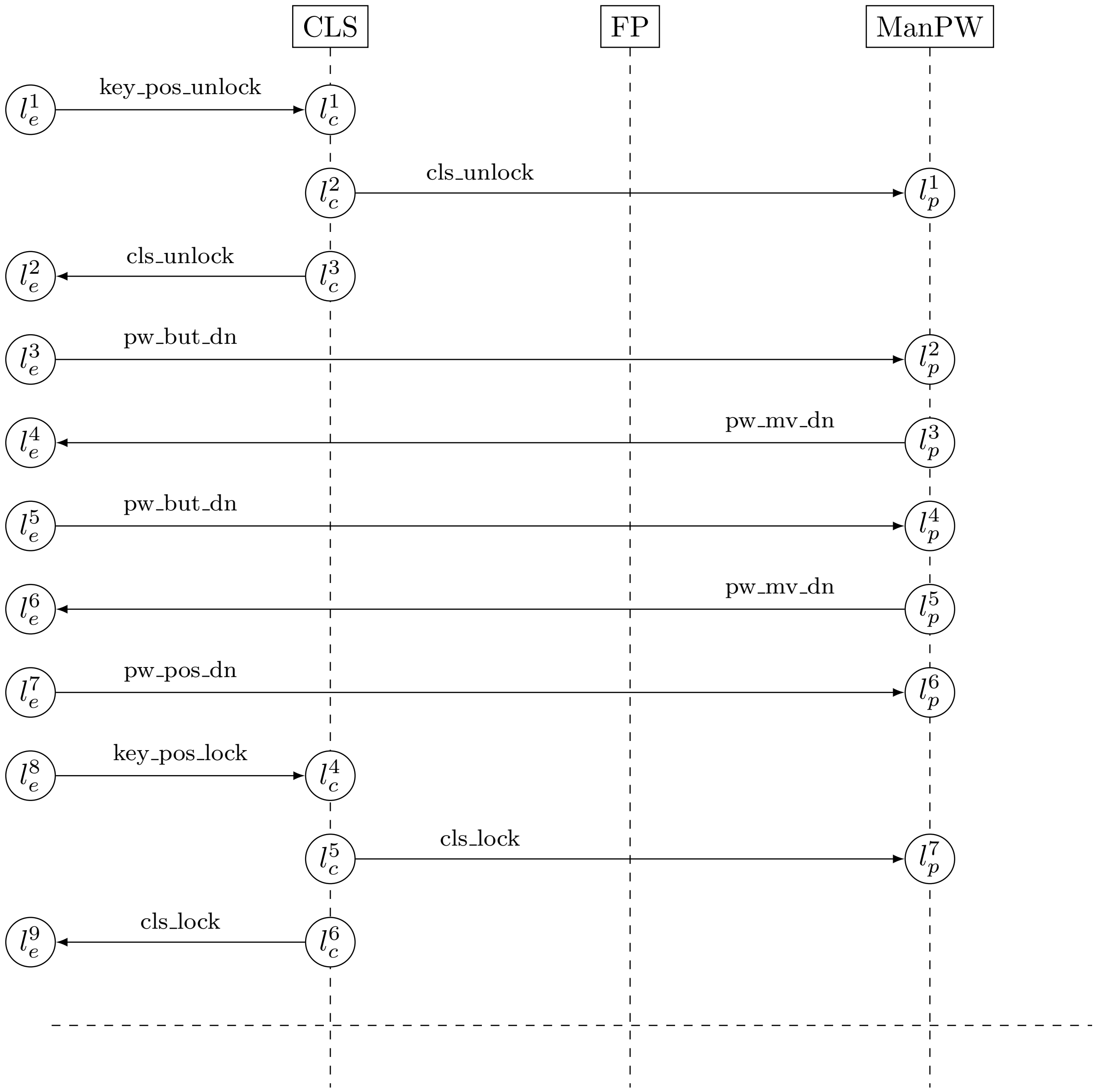

Figure 5.53.: Interaction Test Scenario MSC41 (1)

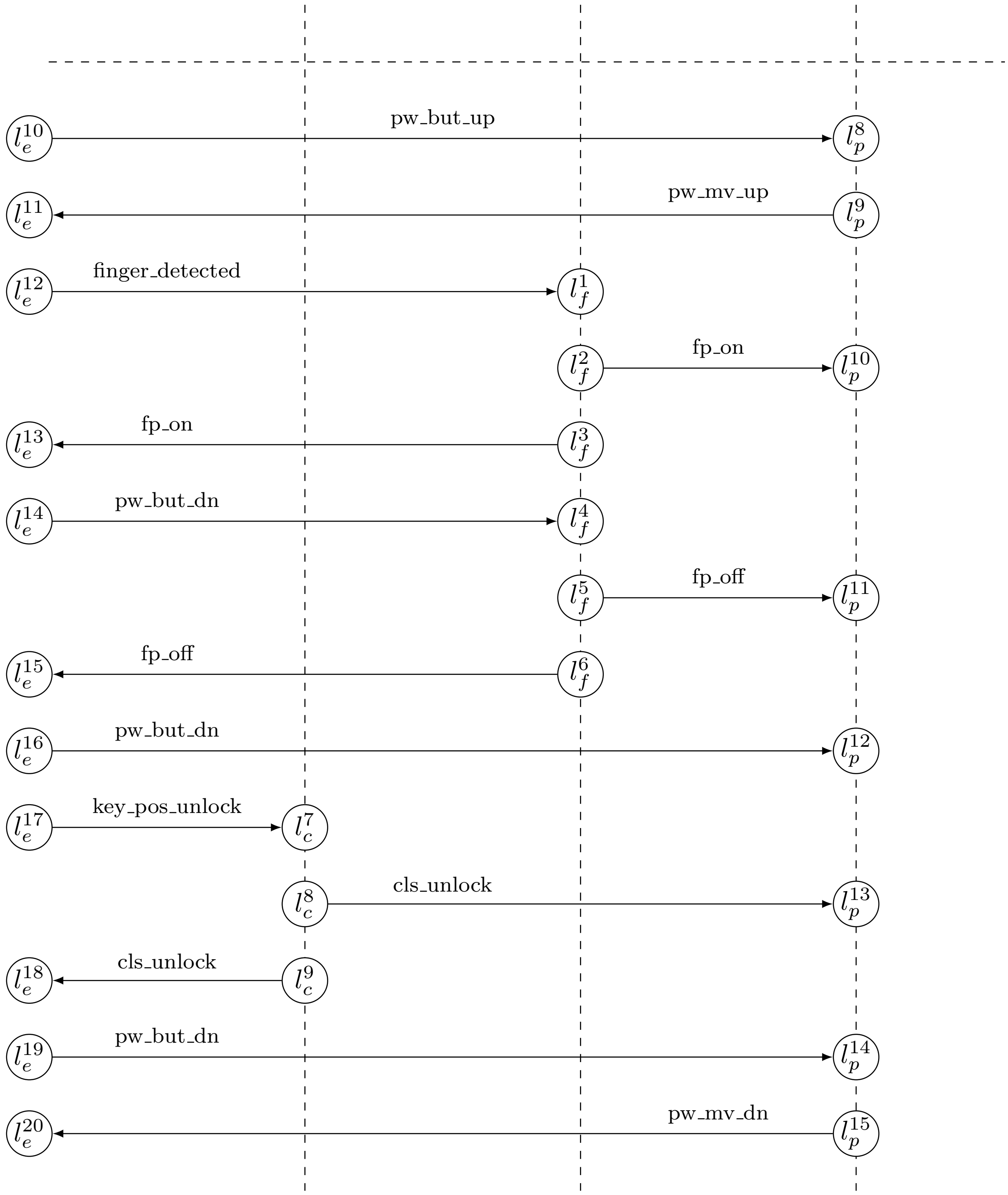

Figure 5.54.: Interaction Test Scenario MSC41 (2)

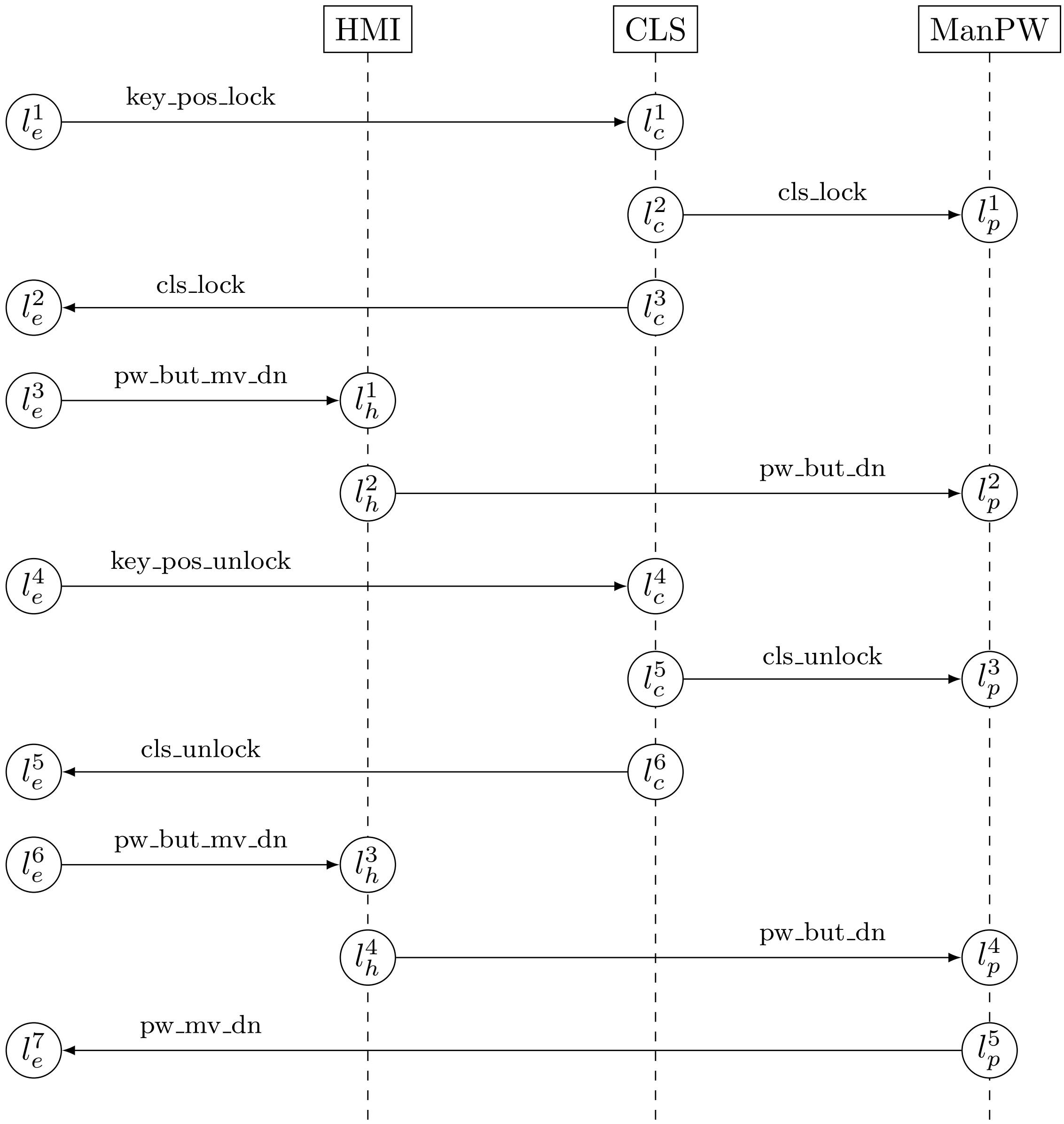

Figure 5.55.: Interaction Test Scenario MSC42

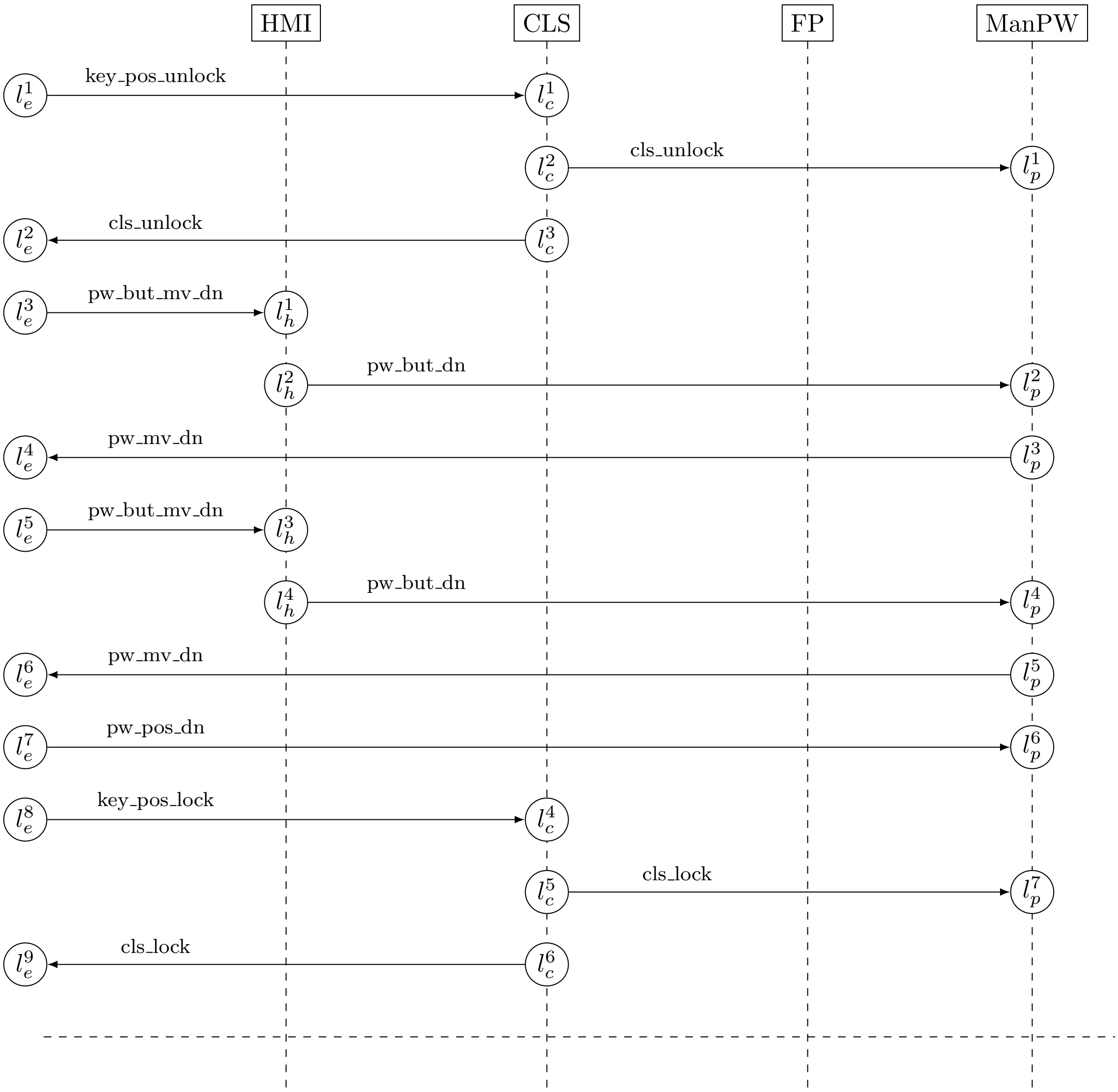

Figure 5.56.: Interaction Test Scenario MSC43 (1)

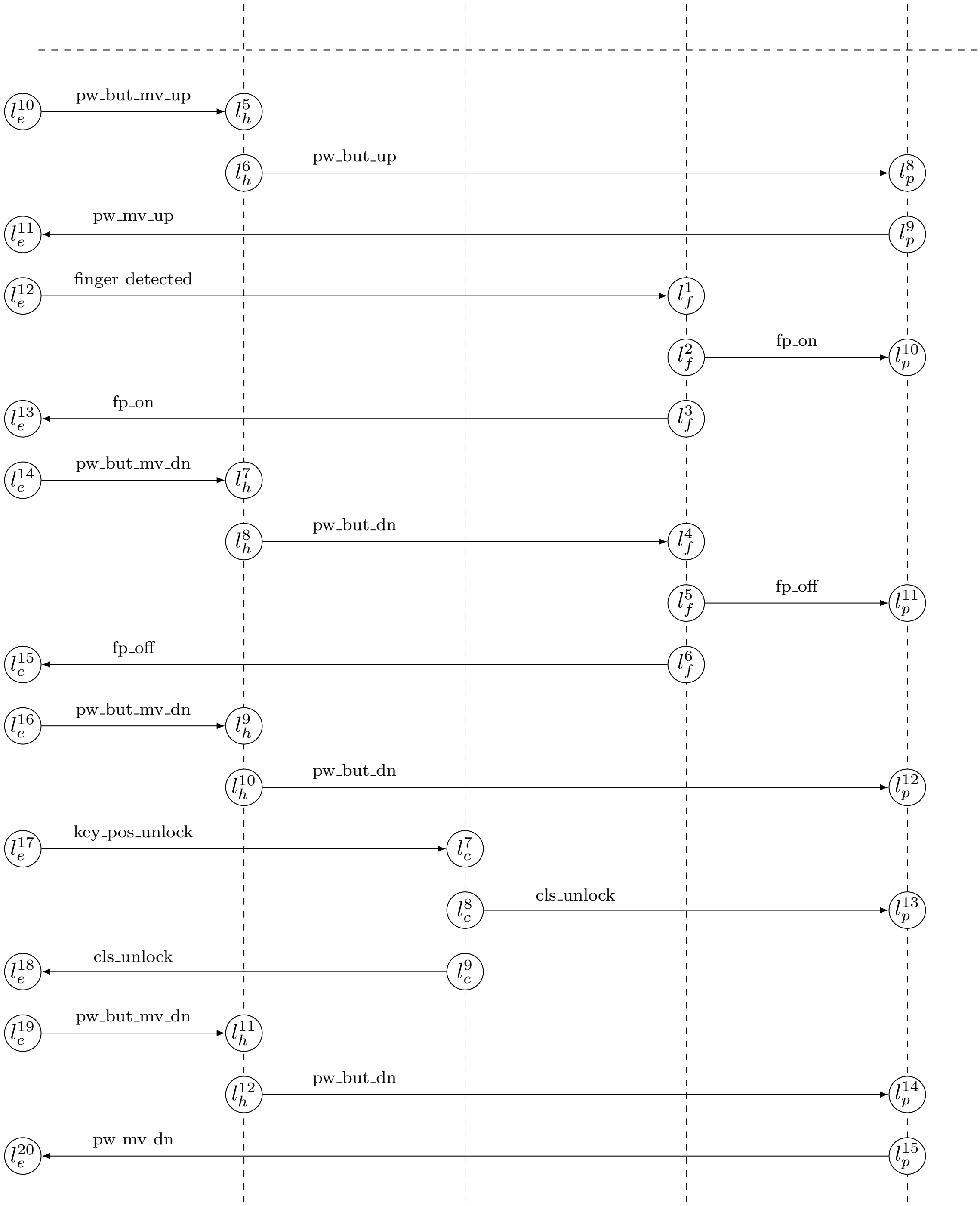

Figure 5.57.: Interaction Test Scenario MSC43 (2)

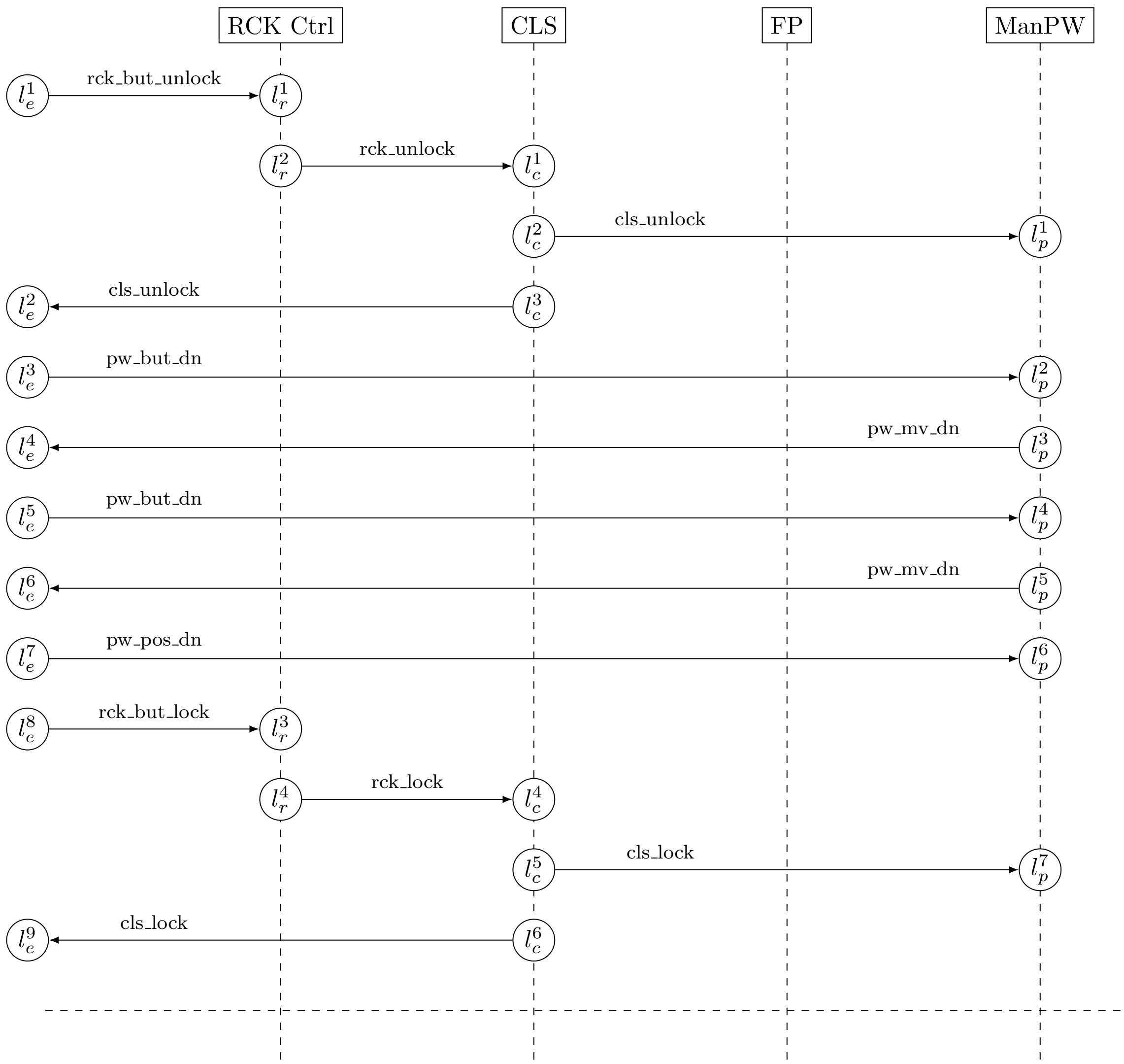

Figure 5.58.: Interaction Test Scenario MSC44 (1)

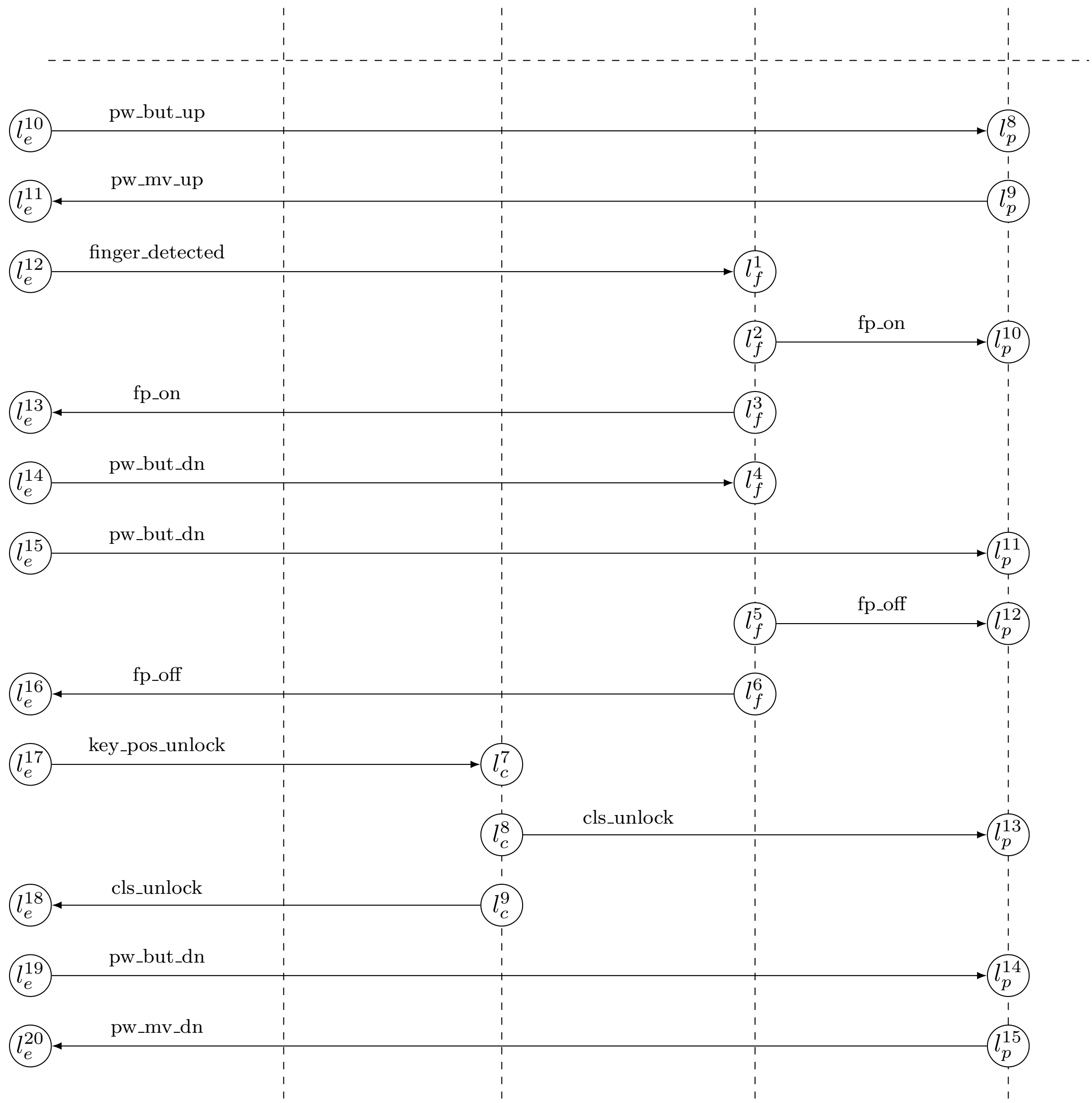

Figure 5.59.: Interaction Test Scenario MSC44 (2)

on/off of the corresponding LED.

Interaction Test Scenario MSC46 The interaction test scenario *MSC46* shown in Fig. 5.61, defines a scenario between the *Manual Power Window* (ManPW) and the *LED Manual Power Window* (LED ManPW). The scenario describes the movement of the manual power window and the turning on/off of the corresponding LEDs. In addition, the scenario defines the blocking of the upwards movement, based on the activated finger protection.

Interaction Test Scenario MSC47 The interaction test scenario *MSC47* defines a scenario between the *Human Machine Interface* (HMI), the *Exterior Mirror with Heating* (EMH), the *LED Exterior Mirror Bottom* (LED EMB), the *LED Exterior Mirror Top* (LED EMT), the *LED Exterior Mirror Right* (LED EMR) and the *LED Exterior Mirror Left* (LED EML). It is depicted in Fig. 5.62, Fig. 5.63 and Fig. 5.64, where the dashed horizontal lines represents a cut for better readability. The scenario describes the movement of the exterior mirror and the turning on/off of the corresponding LEDs if the mirror reaches one of its end positions.

Interaction Test Scenario MSC48 The interaction test scenario *MSC48* shown in Fig. 5.65, defines a scenario between the *Finger Protection* (FP), the *Manual Power Window* (ManPW) and the *LED Manual Power Window* (LED ManPW). The scenario describes the activation of the finger protection and its release by moving the manual power window downwards. In addition, the scenario defines the turning on of the LED for the manual power window, based on its downwards movement.

Interaction Test Scenario MSC49 The interaction test scenario *MSC49* depicted in Fig. 5.66, defines a scenario between the *Finger Protection* (FP), the *LED Finger Protection* (LED FP), the *Manual Power Window* (ManPW) and the *LED Manual Power Window* (LED ManPW). The scenario describes the activation of the finger protection and its release by moving the manual power window downwards. In addition, the scenario defines the turning on of the LED for the manual power window, based on its downwards movement and the turning on/off of the LED for the finger protection, based on its activation/deactivation.

Interaction Test Scenario MSC50 The interaction test scenario *MSC50* shown in Fig. 5.67, defines a scenario between the *Human Machine Interface* (HMI), the *Finger Protection* (FP), the *LED Finger Protection* (LED FP), the *Manual Power Window* (ManPW) and the *LED Manual Power Window* (LED ManPW). The scenario describes the activation of the finger protection and its release by moving the manual power window downwards via the human machine interface. In addition, the scenario defines the turning on of the LED for the manual power window, based on its downwards movement via the human machine interface and the turning on/off of the LED for the finger protection, based on its activation/deactivation.

Interaction Test Scenario MSC51 The interaction test scenario *MSC51* depicted in Fig. 5.68, defines a scenario between the *Human Machine Interface* (HMI), the *Alarm System* (AS) and the *LED Alarm System Interior Monitoring* (LED ASIM). The scenario describes the activation of the alarm system as well as the enabling of the alarm monitoring. In addition, the scenario defines the triggering and stopping of the interior alarm and the turning on/off of the

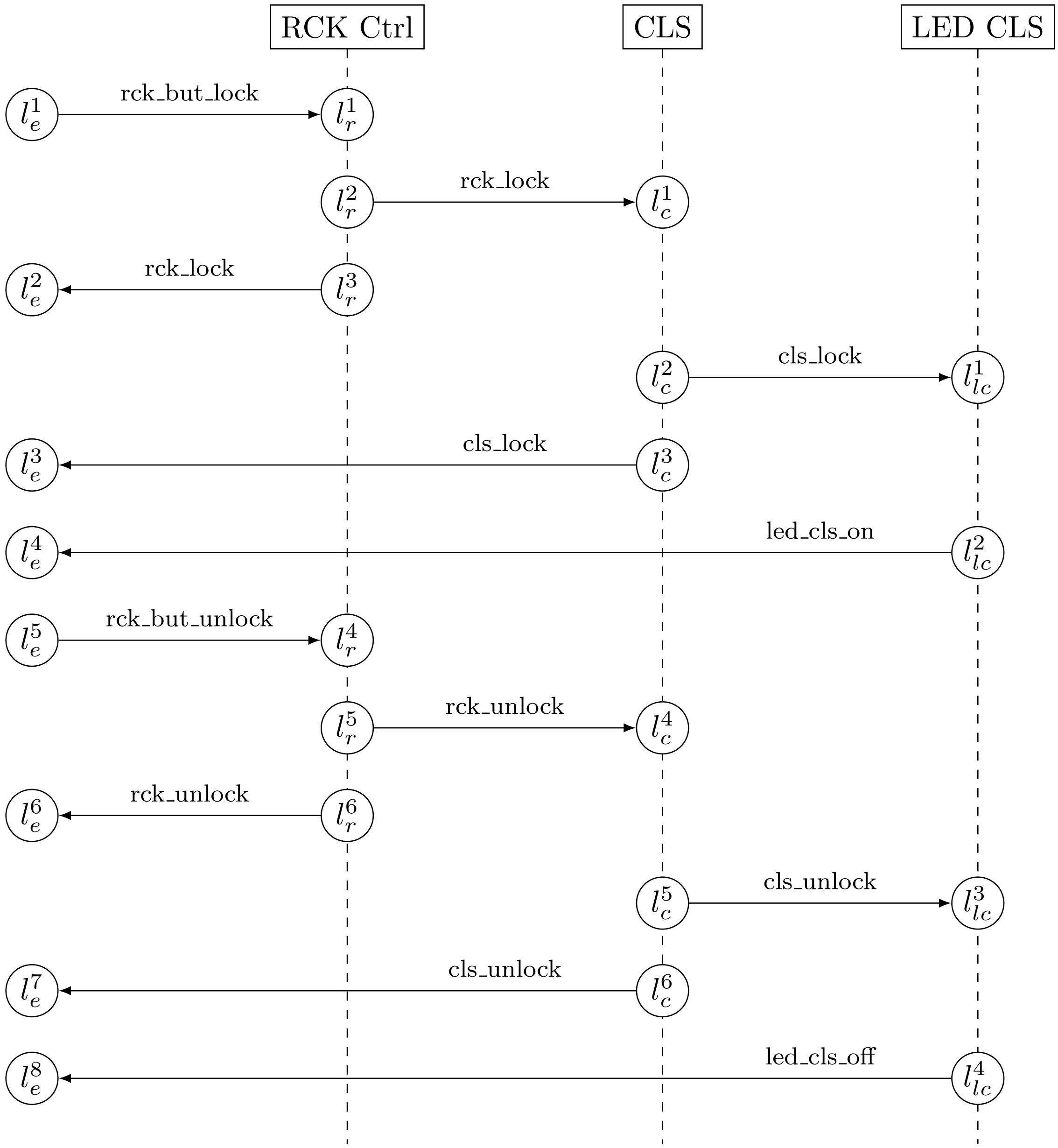

Figure 5.60.: Interaction Test Scenario MSC45

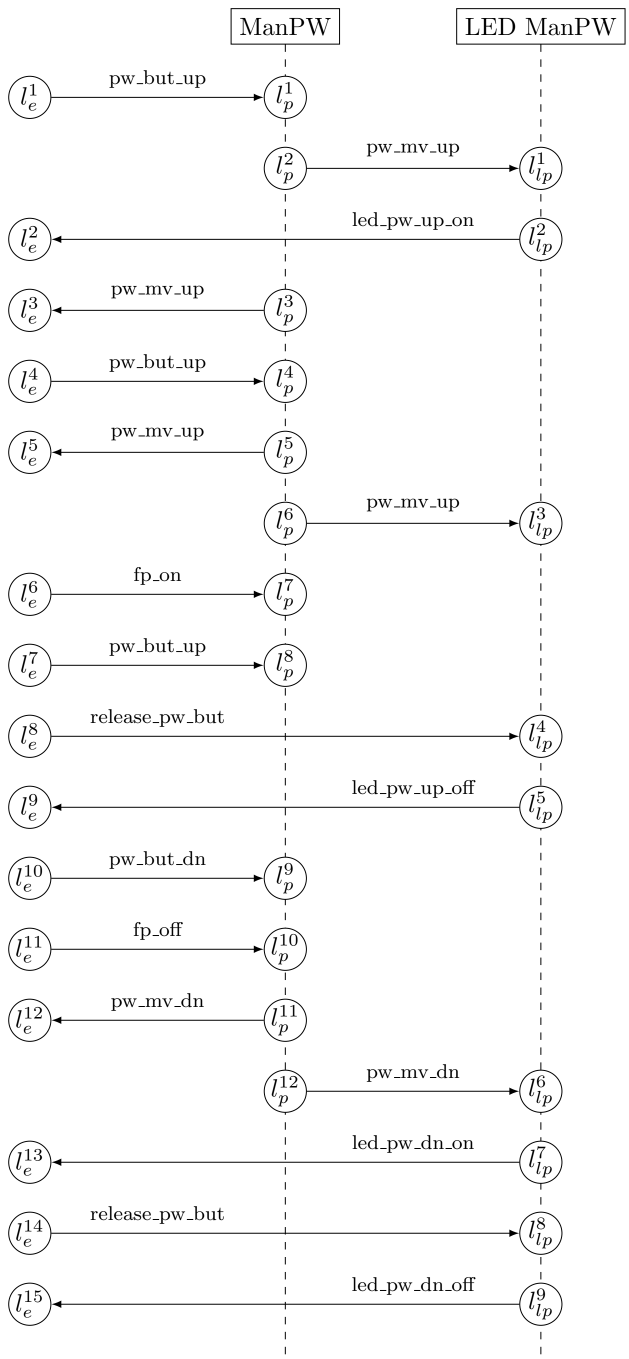

Figure 5.61.: Interaction Test Scenario MSC46

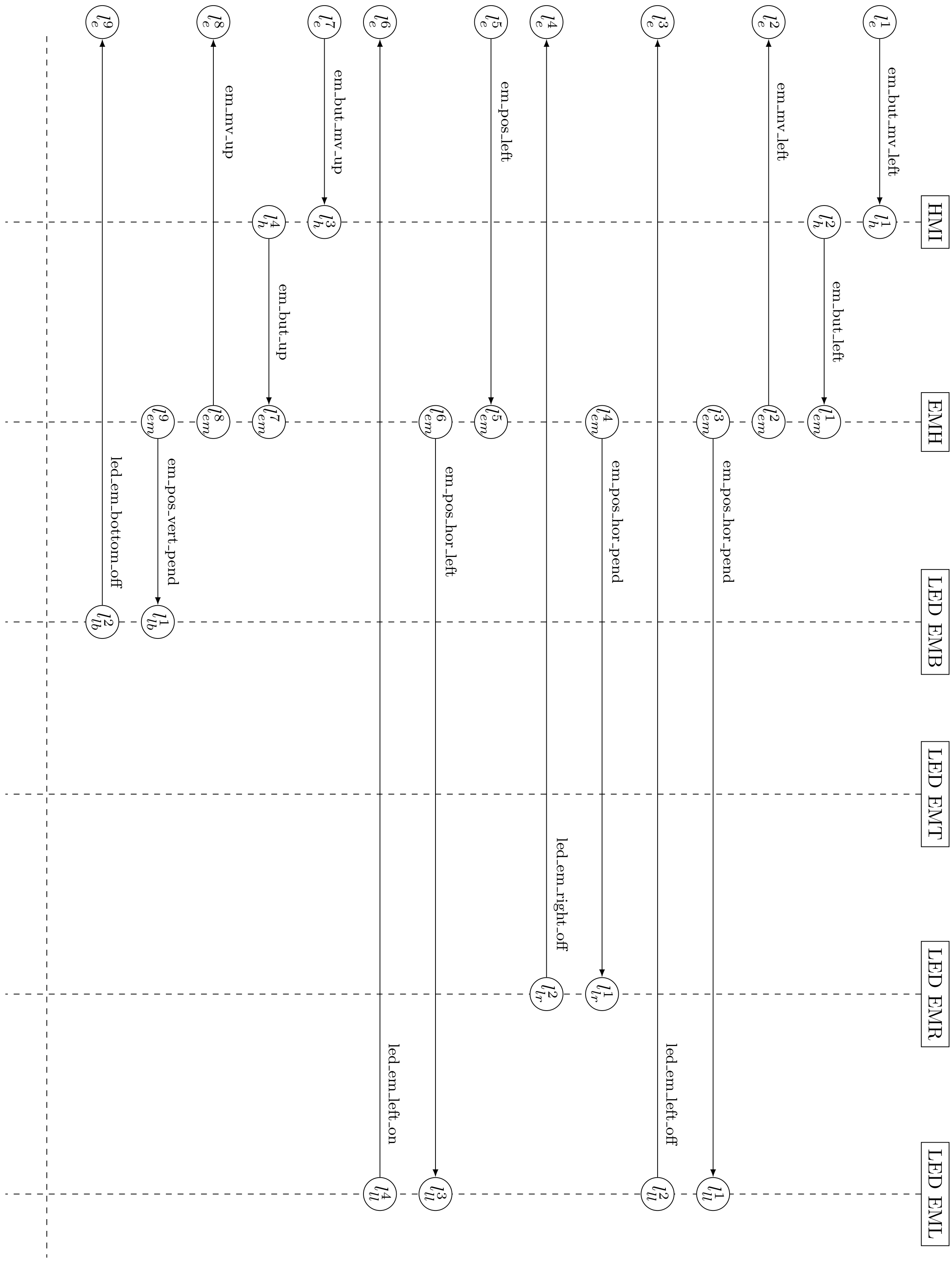

Figure 5.62.: Interaction Test Scenario MSC47 (1)

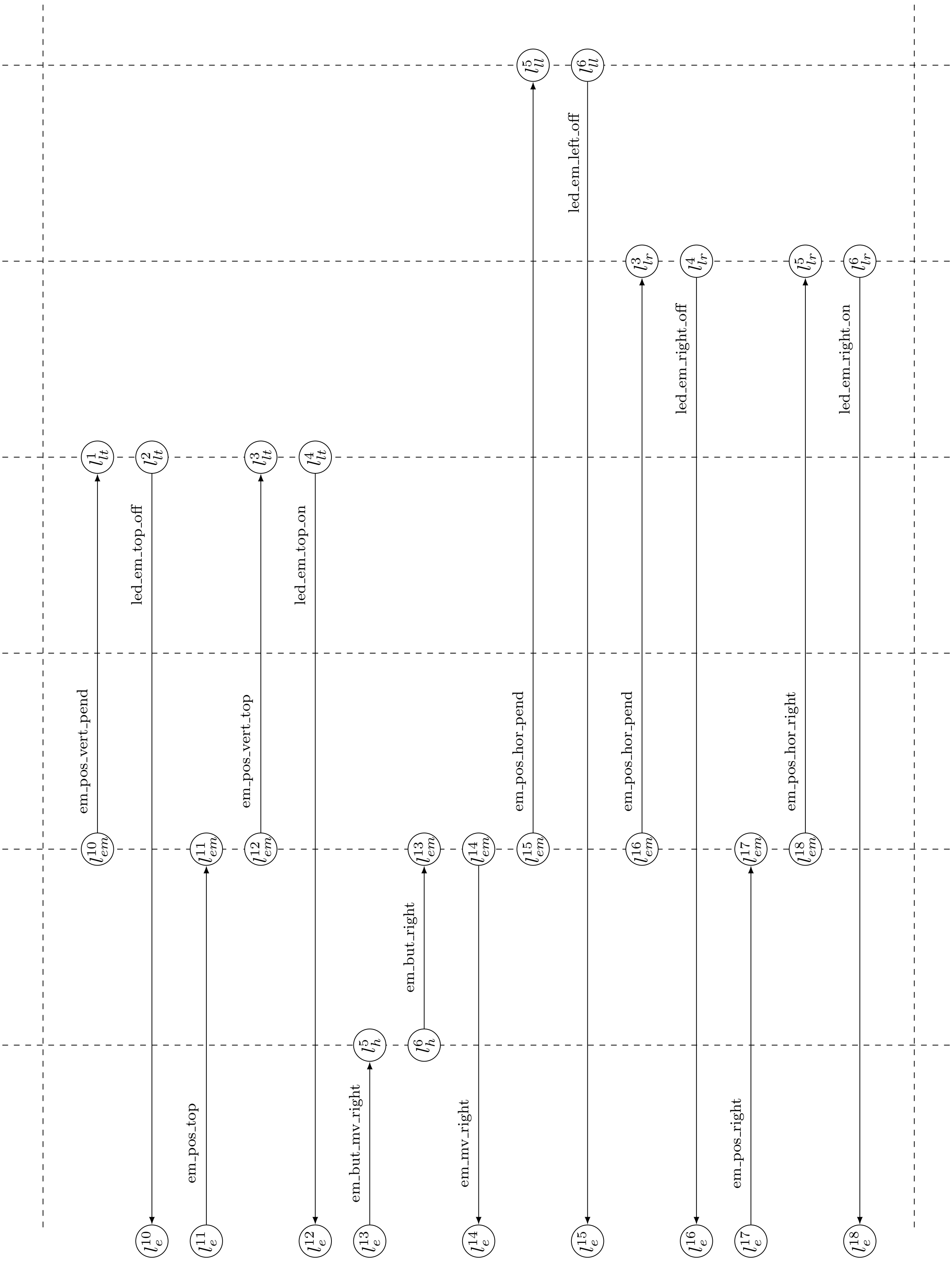

Figure 5.63.: Interaction Test Scenario MSC47 (2)

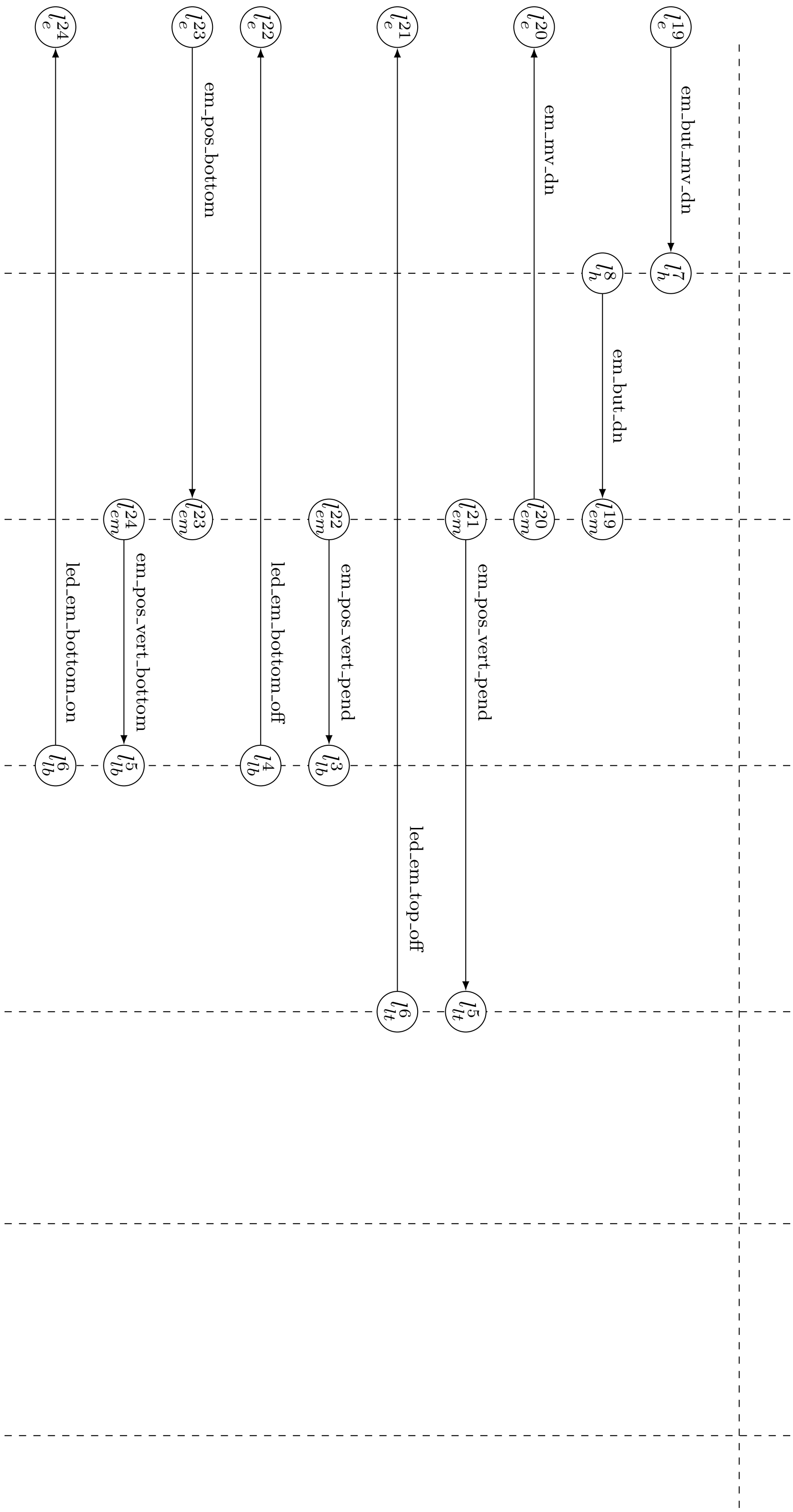

Figure 5.64.: Interaction Test Scenario MSC47 (3)

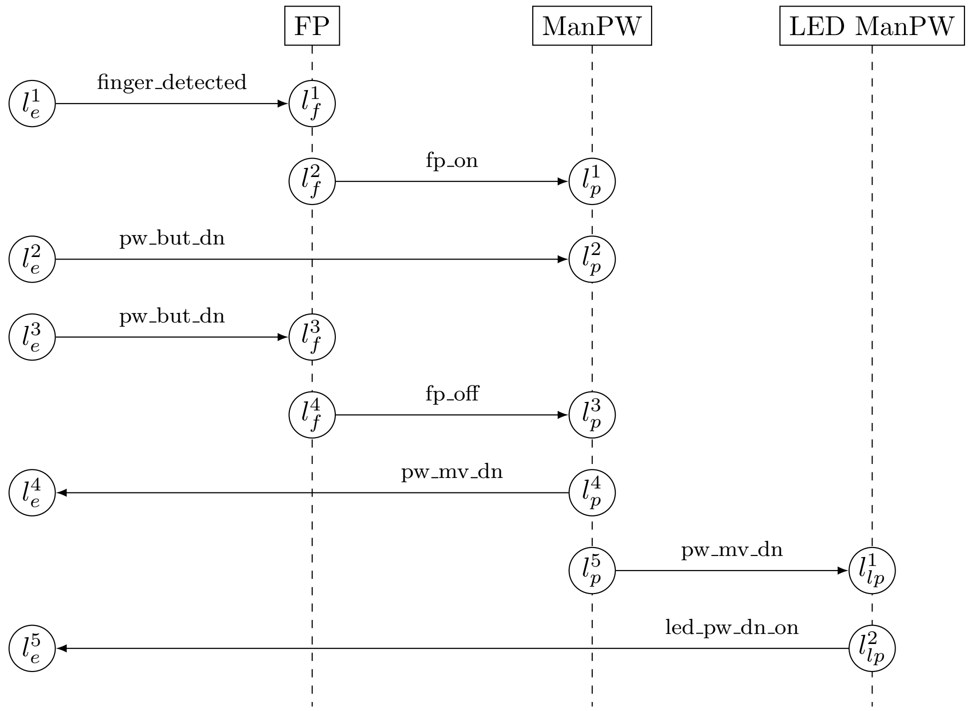

Figure 5.65.: Interaction Test Scenario MSC48

corresponding LED.

Interaction Test Scenario MSC52 The interaction test scenario *MSC52* shown in Fig. 5.69, defines a scenario between the *Human Machine Interface* (HMI), the *Alarm System* (AS), the *LED Alarm System Interior Monitoring* (LED ASIM) and the *LED Alarm System Alarm Active* (LED ASAC). The scenario describes the activation of the alarm system as well as the enabling of the alarm monitoring and the turning on/off of the corresponding LED. In addition, the scenario defines the triggering and stopping of the interior alarm and the turning on/off of the corresponding LED as well as the turning off of the LED for the alarm monitoring.

Interaction Test Scenario MSC53 The interaction test scenario *MSC53* depicted in Fig. 5.70, defines a scenario between the *Finger Protection* (FP), the *LED Finger Protection* (LED FP) and the *Manual Power Window* (ManPW). The scenario describes the activation and deactivation of the finger protection and the turning on/off of the corresponding LED.

Interaction Test Scenario MSC54 The interaction test scenario *MSC54* defines a scenario between the *Automatic Power Window* (AutoPW) and the *LED Automatic Power Window* (LED AutoPW). It is shown in Fig. 5.71 and Fig. 5.72, where the dashed horizontal line represents a cut for better readability. The scenario describes the automated upwards and downwards movement of the automatic power window to its corresponding end positions and the turning on/off of the LED for the upwards movement and of the LED for the downwards movement.

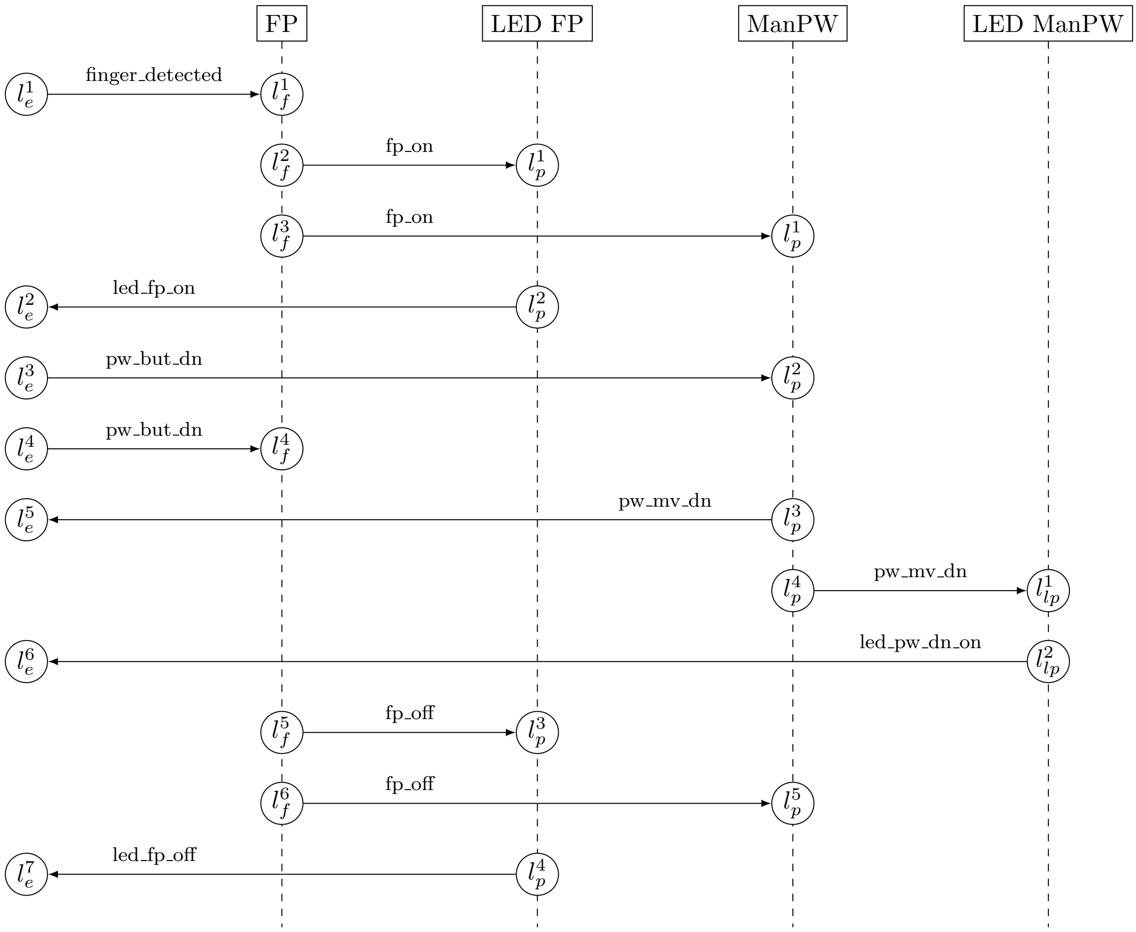

Figure 5.66.: Interaction Test Scenario MSC49

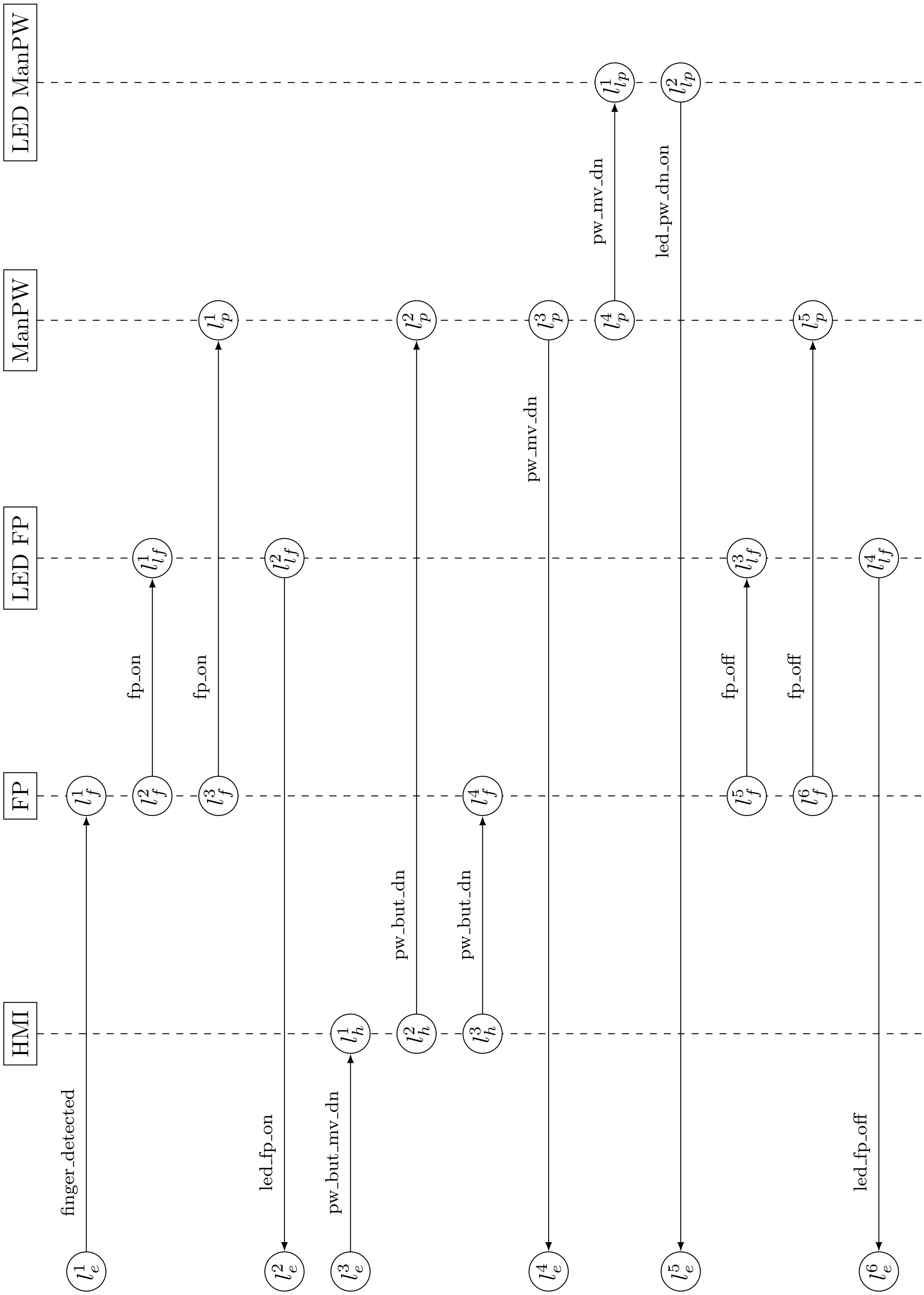

Figure 5.67: Interaction Test Scenario MSC50

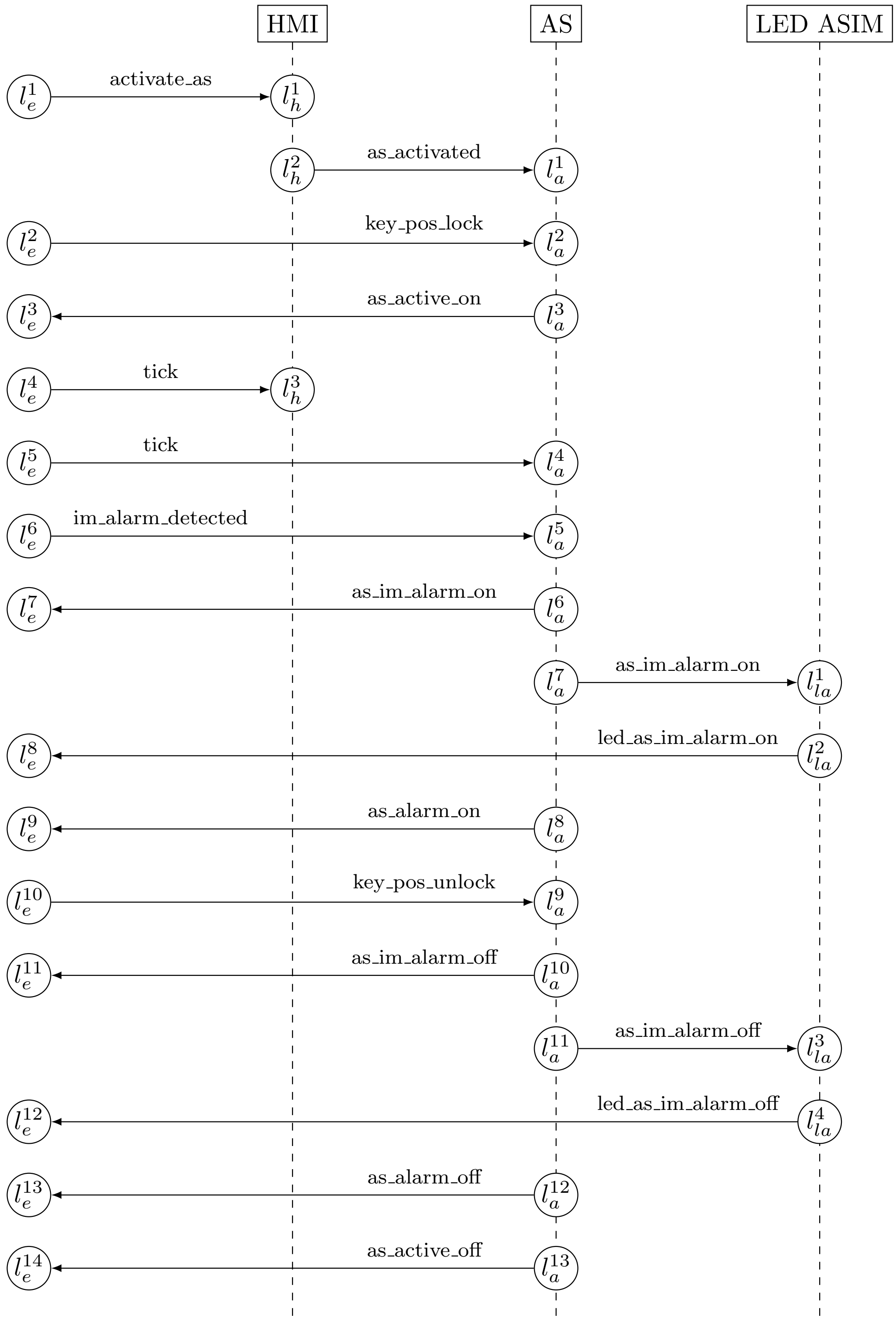

Figure 5.68.: Interaction Test Scenario MSC51

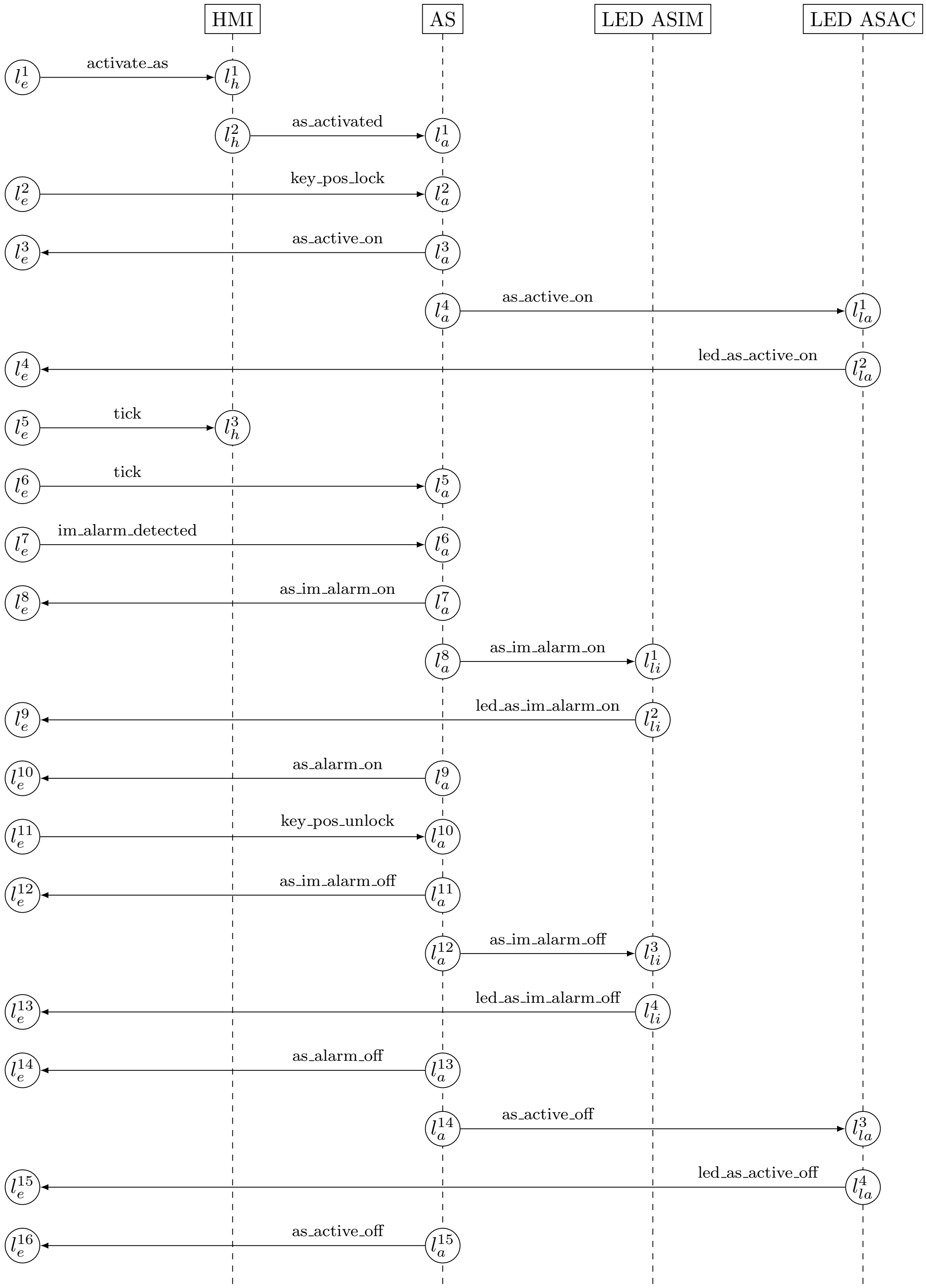

Figure 5.69.: Interaction Test Scenario MSC52

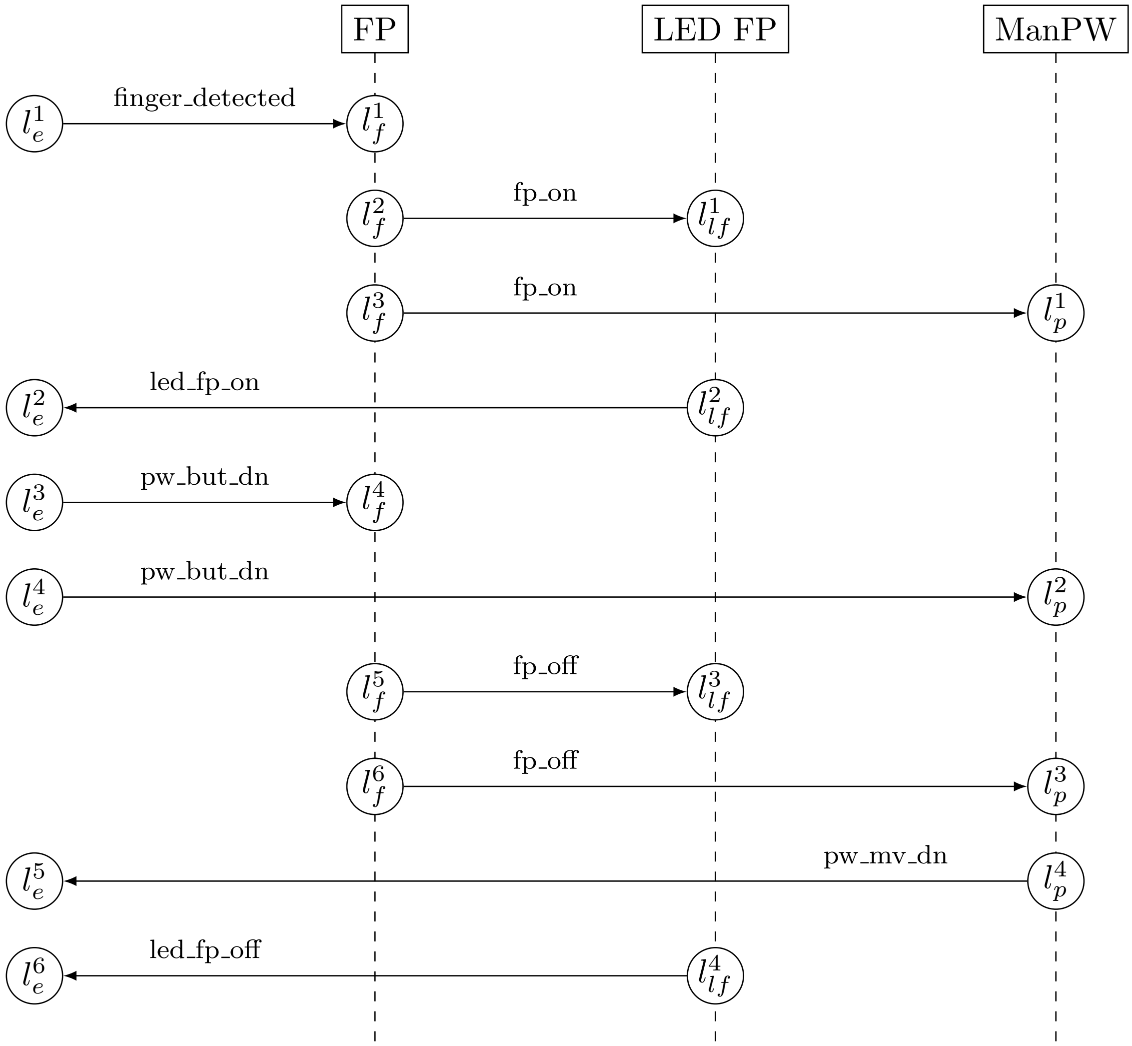

Figure 5.70.: Interaction Test Scenario MSC53

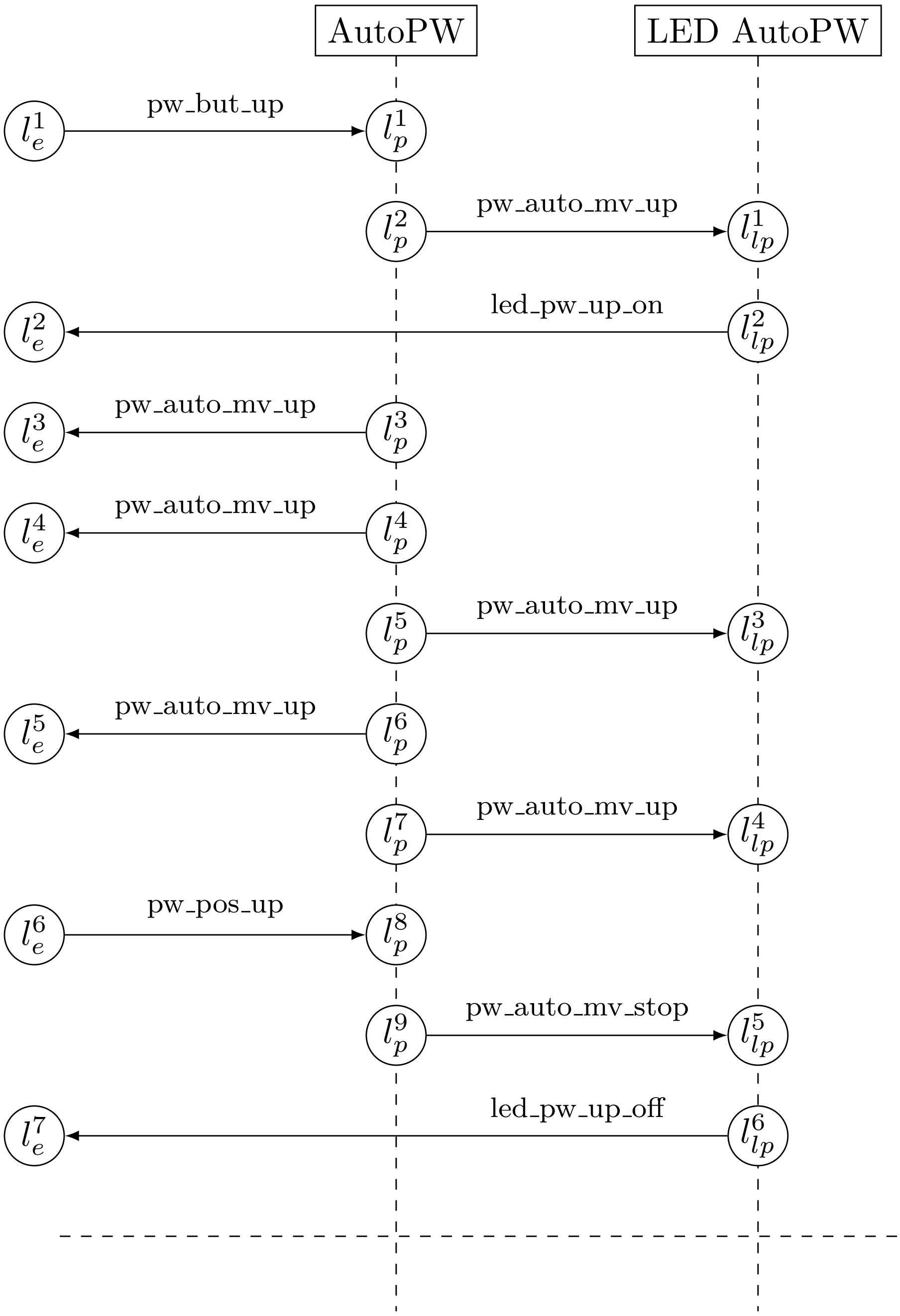

Figure 5.71.: Interaction Test Scenario MSC54 (1)

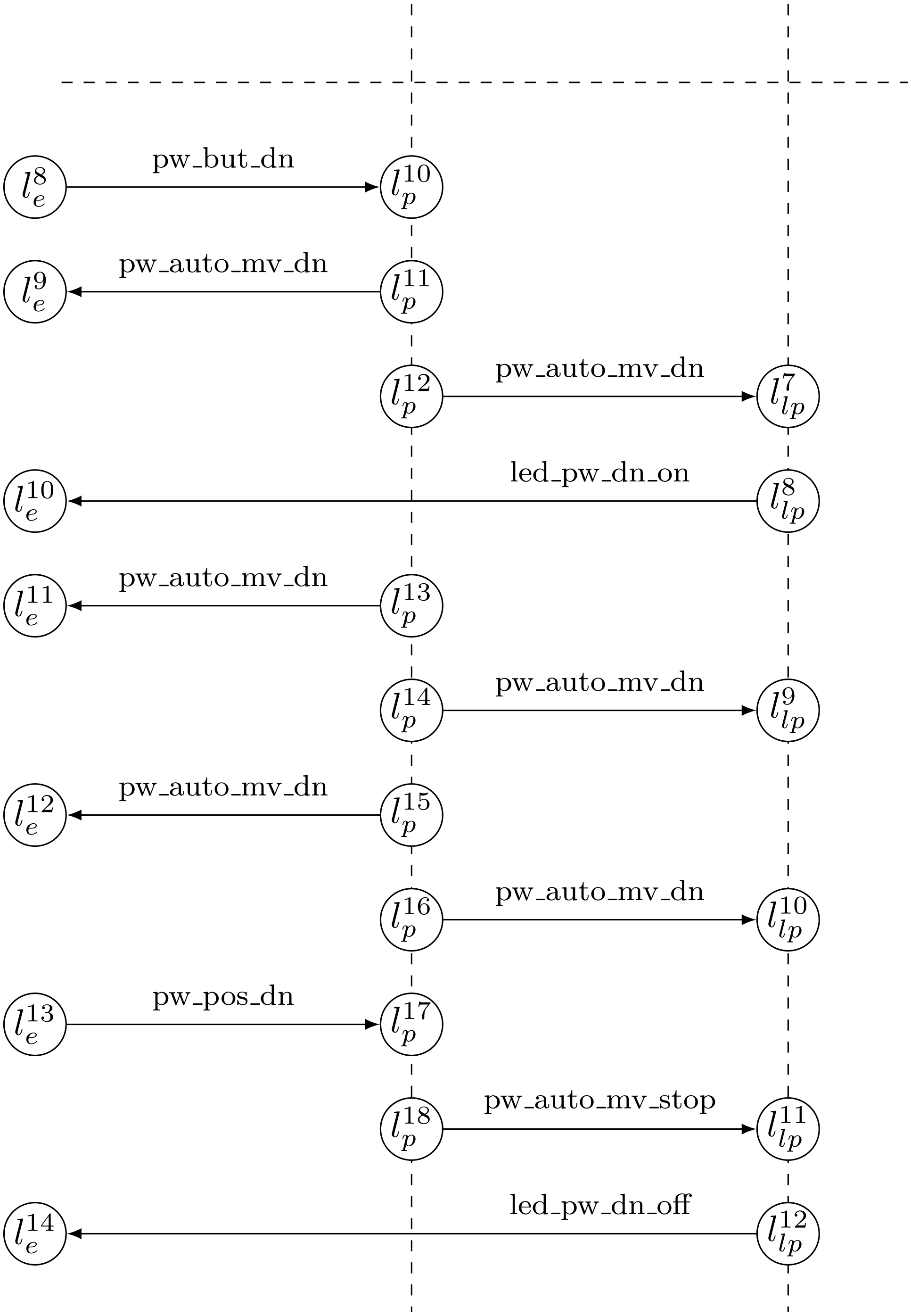

Figure 5.72.: Interaction Test Scenario MSC54 (2)

Interaction Test Scenario MSC55 The interaction test scenario *MSC55* defines a scenario between the *Human Machine Interface* (HMI), the *Automatic Power Window* (AutoPW) and the *LED Automatic Power Window* (LED AutoPW). It is depicted in Fig. 5.73 and Fig. 5.74, where the dashed horizontal line represents a cut for better readability. The scenario describes the automated upwards and downwards movement of the automatic power window to its corresponding end positions via the human machine interface and the turning on/off of the LED for the upwards movement and of the LED for the downwards movement.

Interaction Test Scenario MSC56 The interaction test scenario *MSC56* defines a scenario between the *Human Machine Interface* (HMI), the *Finger Protection* (FP), the *Automatic Power Window* (AutoPW) and the *LED Automatic Power Window* (LED AutoPW). It is depicted in Fig. 5.75 and Fig. 5.76, where the dashed horizontal line represents a cut for better readability. The scenario describes the automated upwards and downwards movement of the automatic power window to its corresponding end positions via the human machine interface and the turning on/off of the LED for the upwards movement and of the LED for the downwards movement. In addition, the scenario defines the blocking of the automated upwards movement, based on the activation of the finger protection.

Interaction Test Scenario MSC57 The interaction test scenario *MSC57* defines a scenario between the *Finger Protection* (FP), the *Automatic Power Window* (AutoPW) and the *LED Automatic Power Window* (LED AutoPW). It is shown in Fig. 5.77 and Fig. 5.78, where the dashed horizontal line represents a cut for better readability. The scenario describes the automated upwards and downwards movement of the automatic power window to its corresponding end positions and the turning on/off of the LED for the upwards movement and of the LED for the downwards movement. In addition, the scenario defines the blocking of the automated upwards movement, based on the activation of the finger protection.

Interaction Test Scenario MSC58 The interaction test scenario *MSC58* depicted in Fig. 5.79, defines a scenario between the *Finger Protection* (FP), the *LED Finger Protection* (LED FP), the *Automatic Power Window* (AutoPW) and the *LED Automatic Power Window* (LED AutoPW). The scenario describes the activation and deactivation of the finger protection and the turning on/off of the corresponding LEDs for the finger protection and for the downwards movement of the automatic power window.

Interaction Test Scenario MSC59 The interaction test scenario *MSC59* shown in Fig. 5.80, defines a scenario between the *Human Machine Interface* (HMI), the *Finger Protection* (FP), the *LED Finger Protection* (LED FP), the *Automatic Power Window* (AutoPW) and the *LED Automatic Power Window* (LED AutoPW). The scenario describes the activation of the finger protection and its deactivation, based on the downwards movement of the window initiated via the human machine interface. In addition, the scenario defines the turning on/off of the corresponding LEDs for the finger protection and for the downwards movement of the automatic power window.

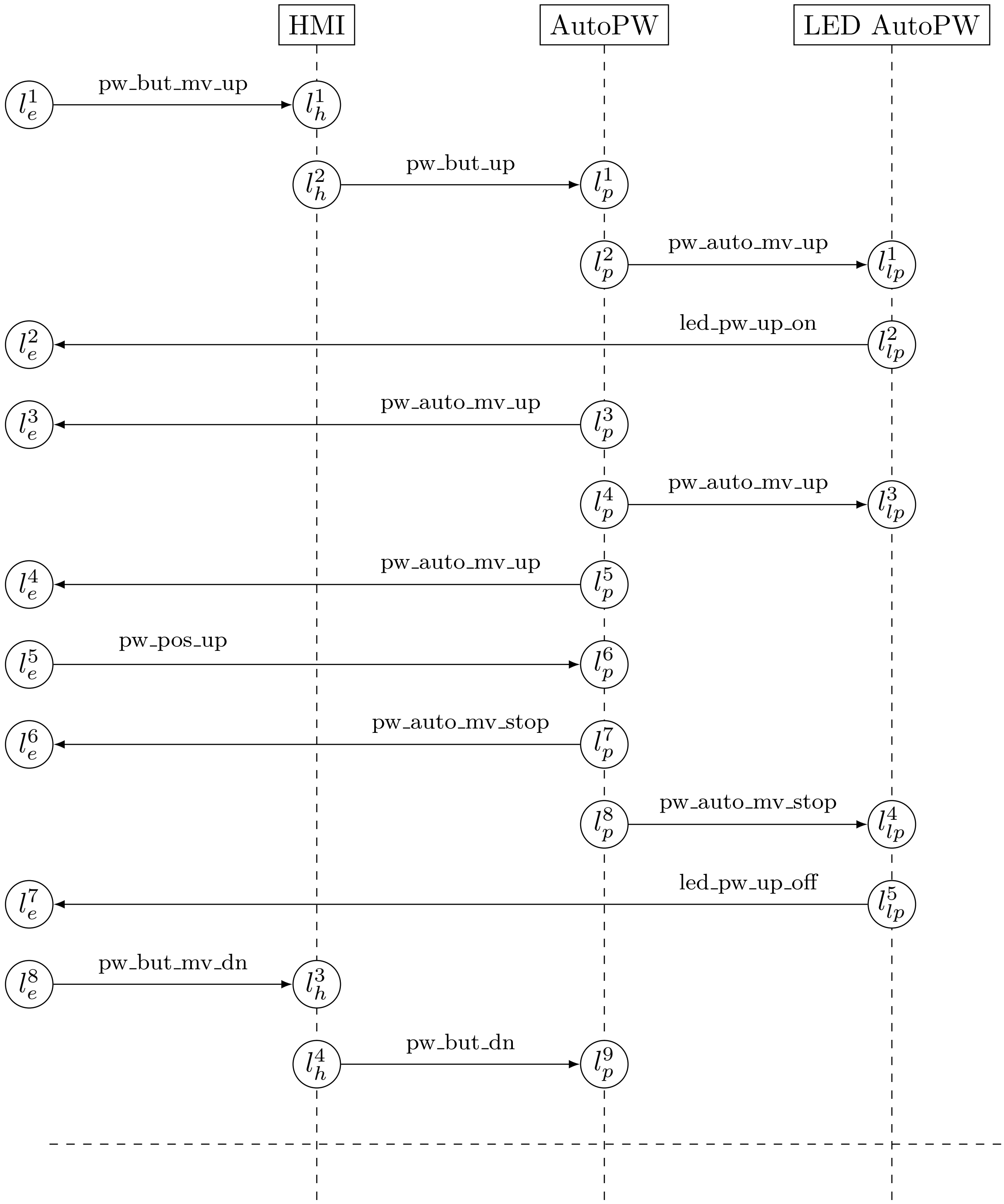

Figure 5.73.: Interaction Test Scenario MSC55 (1)

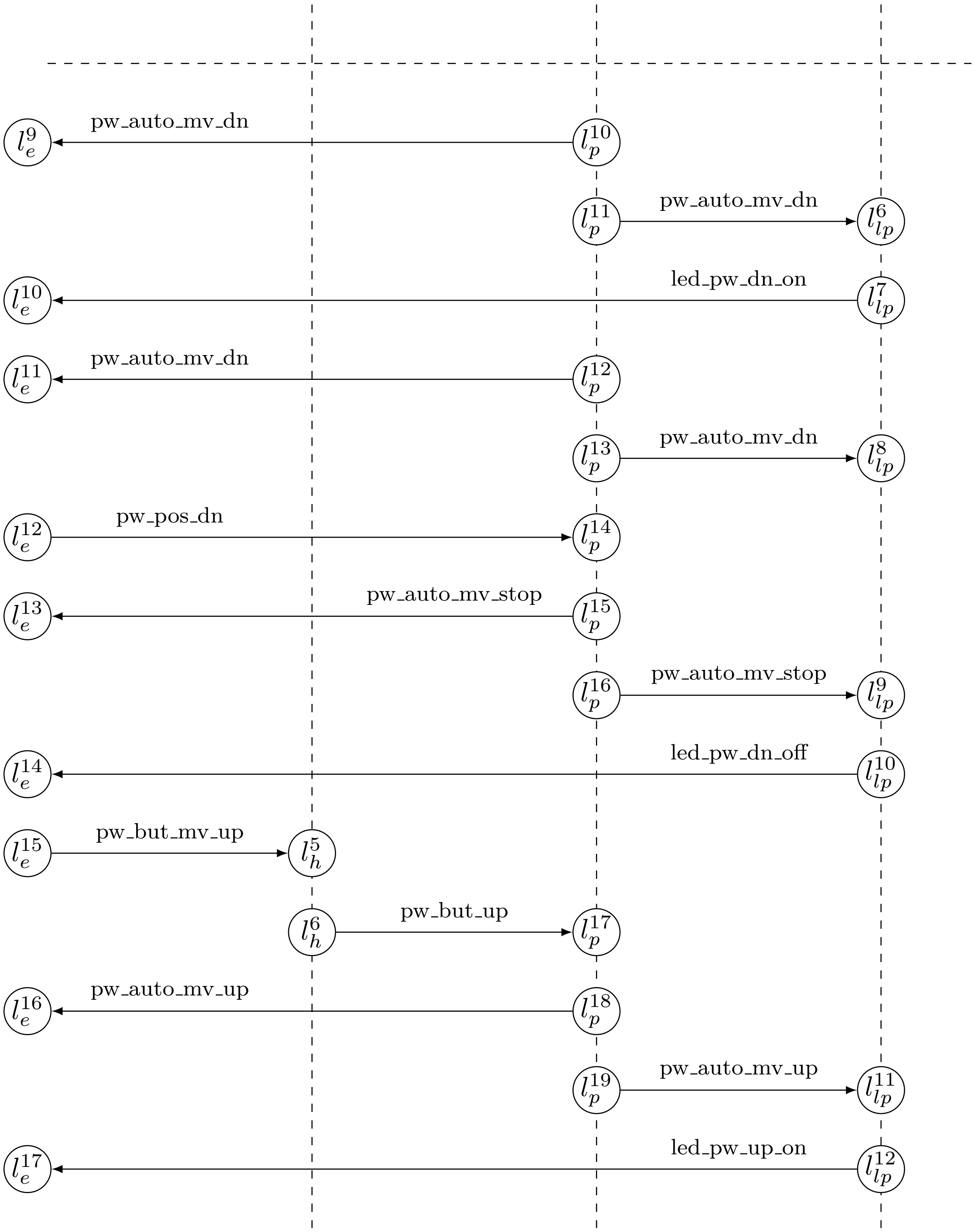

Figure 5.74.: Interaction Test Scenario MSC55 (2)

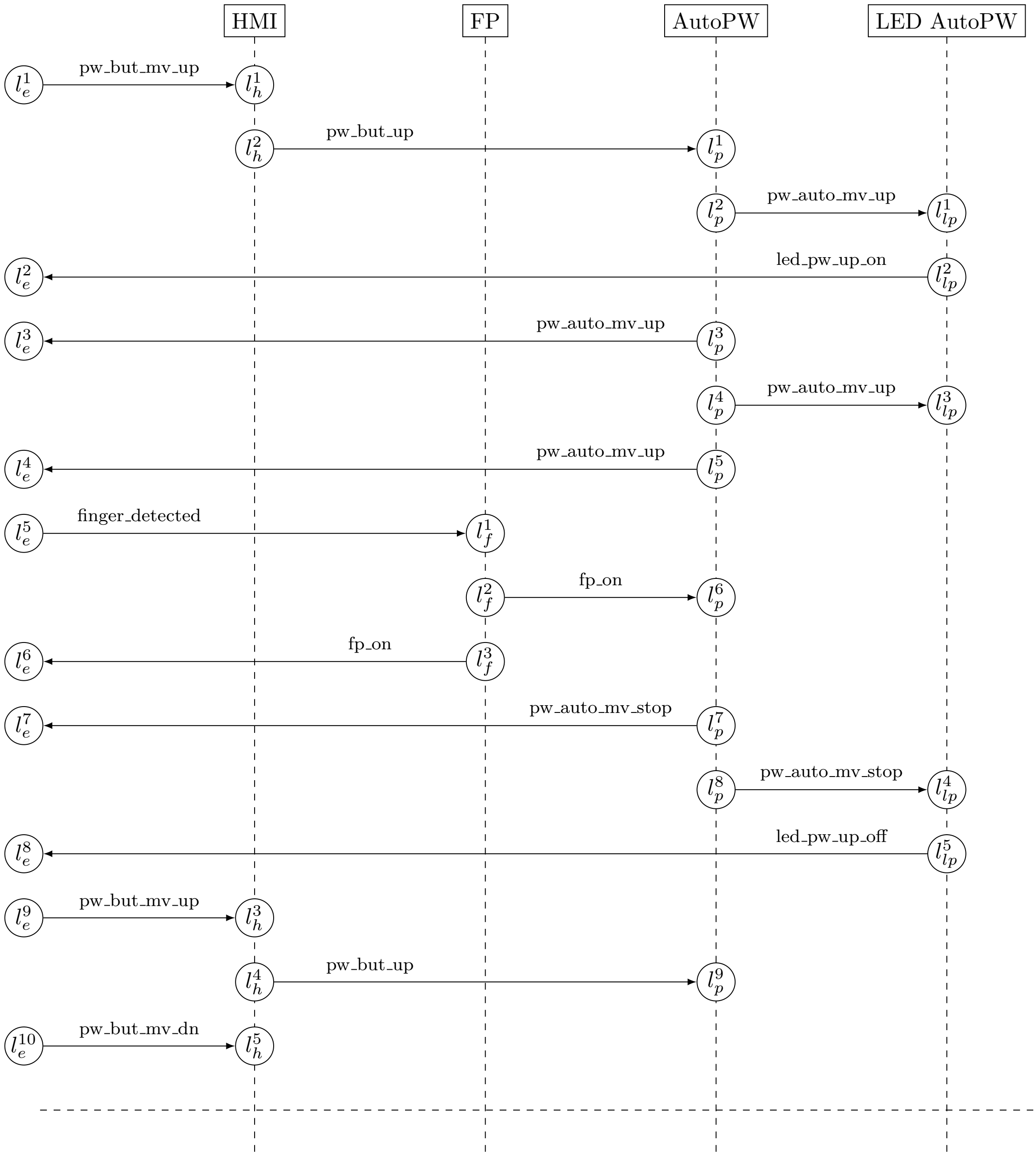

Figure 5.75.: Interaction Test Scenario MSC56 (1)

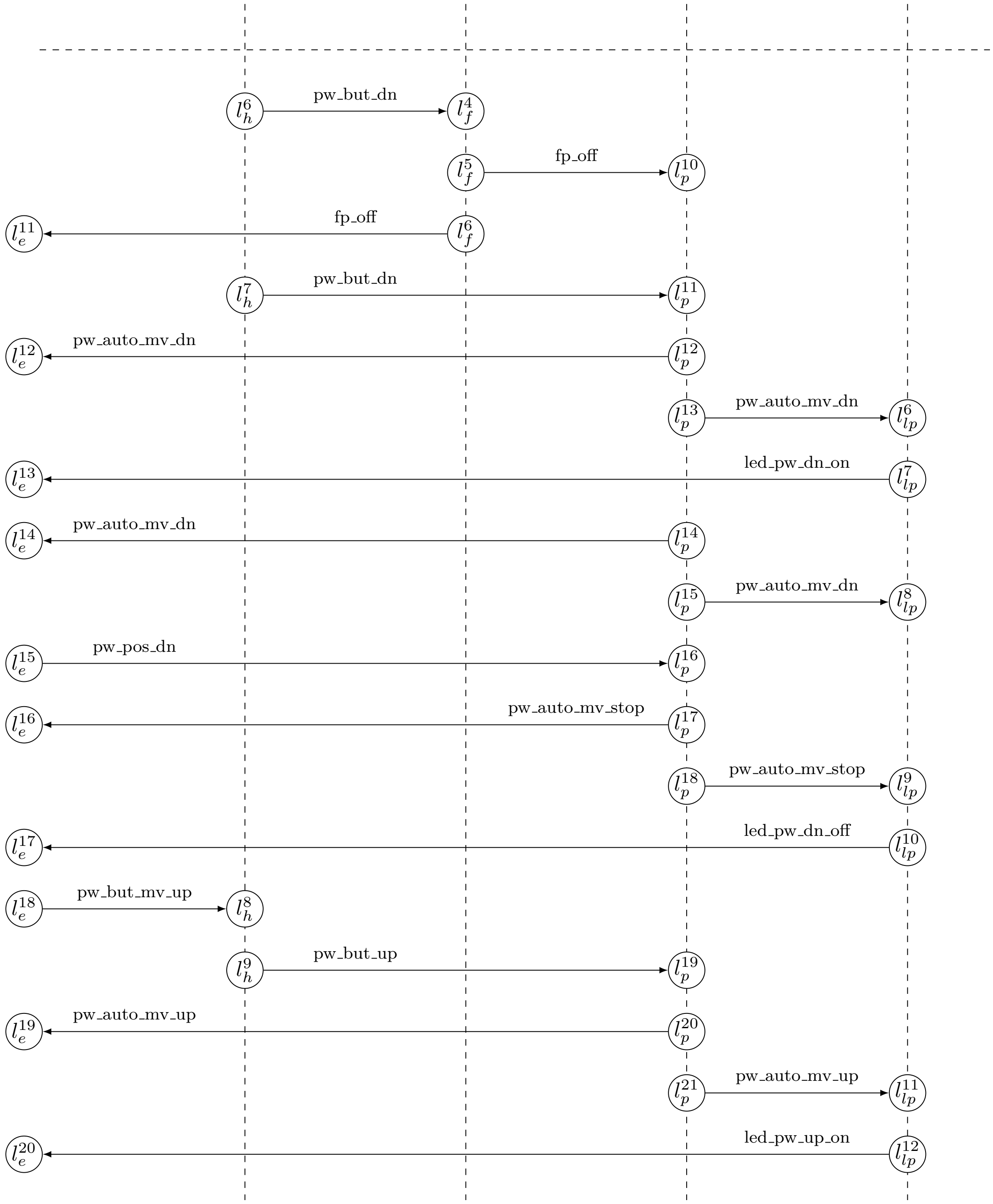

Figure 5.76.: Interaction Test Scenario MSC56 (2)

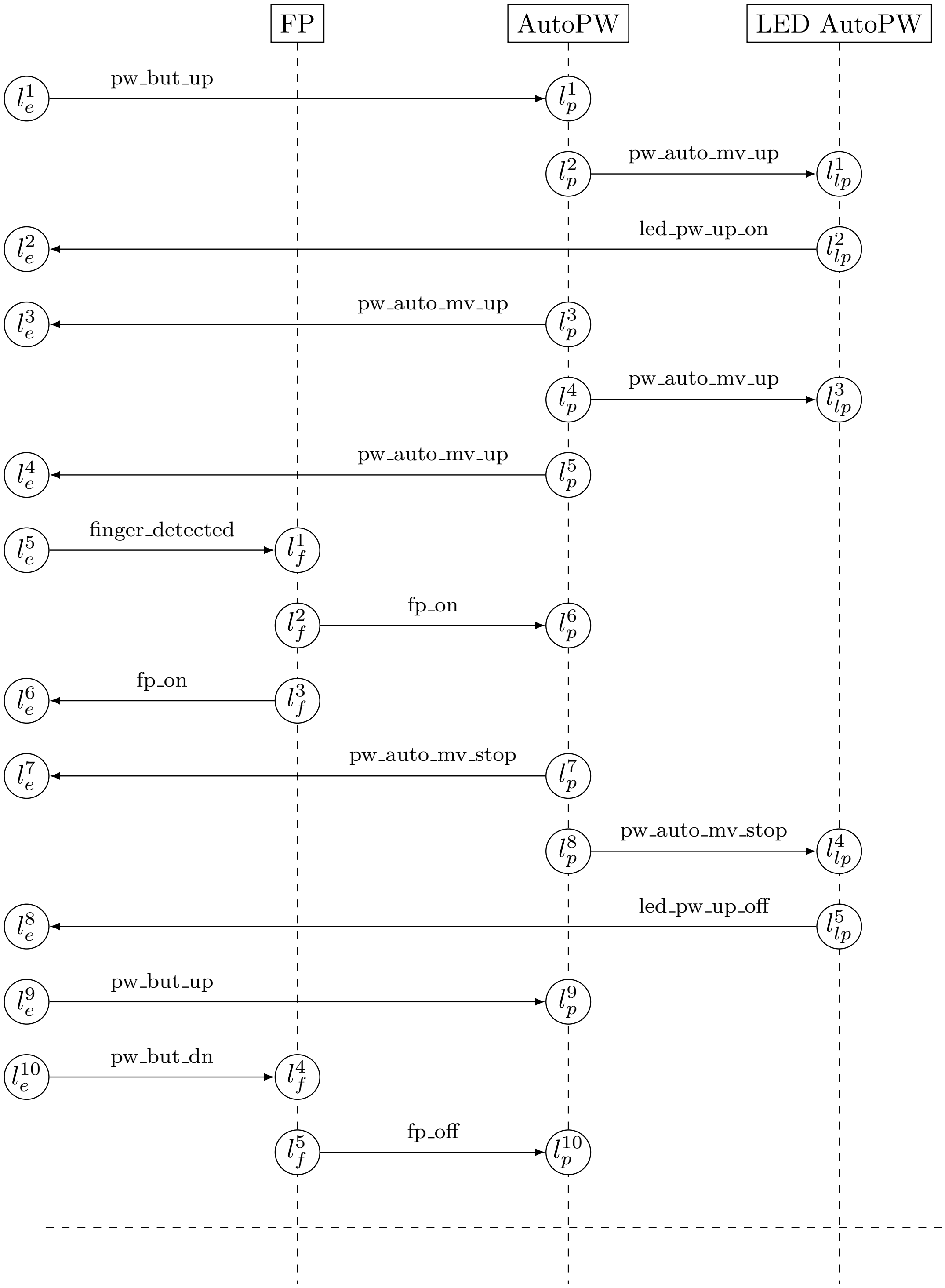

Figure 5.77.: Interaction Test Scenario MSC57 (1)

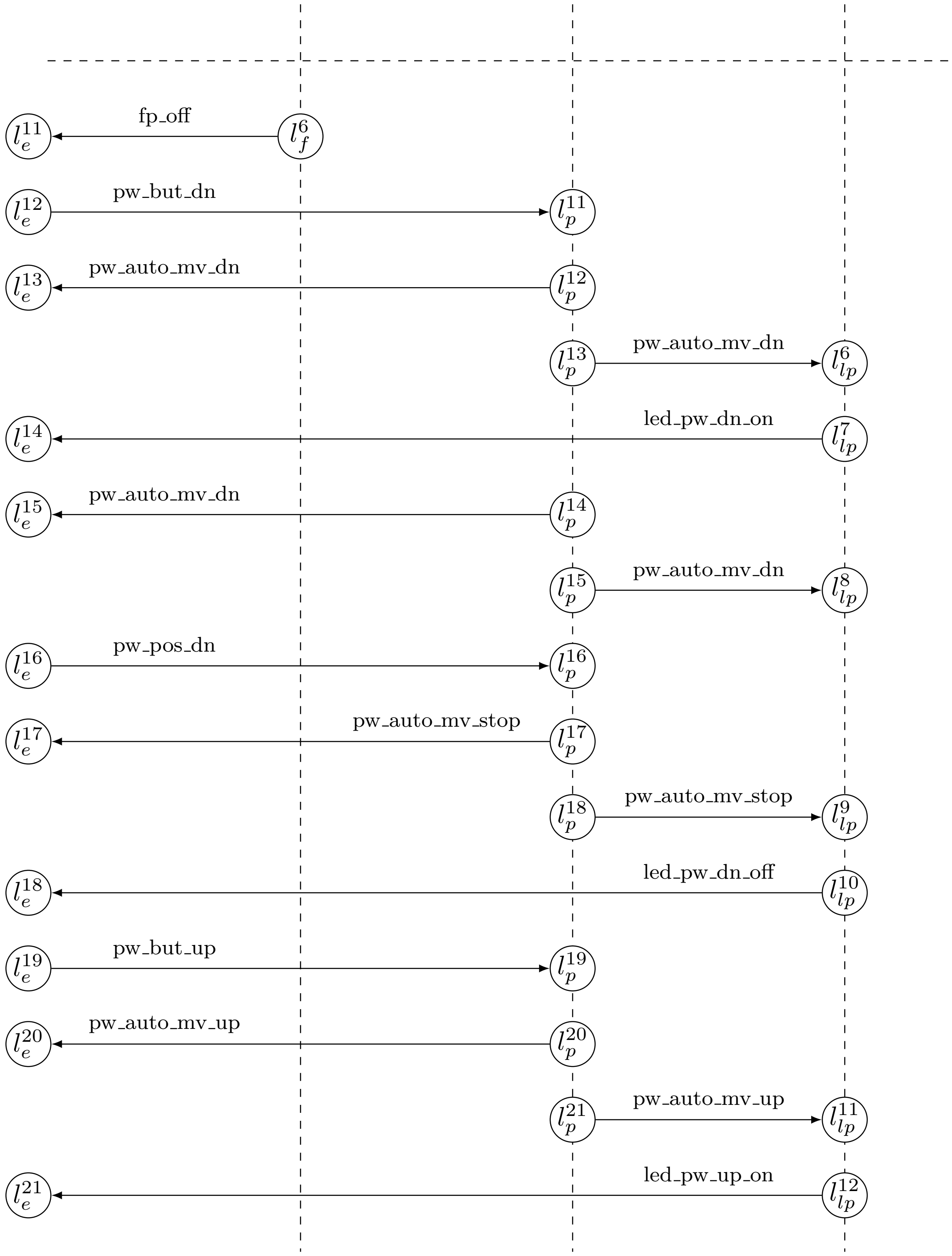

Figure 5.78.: Interaction Test Scenario MSC57 (2)

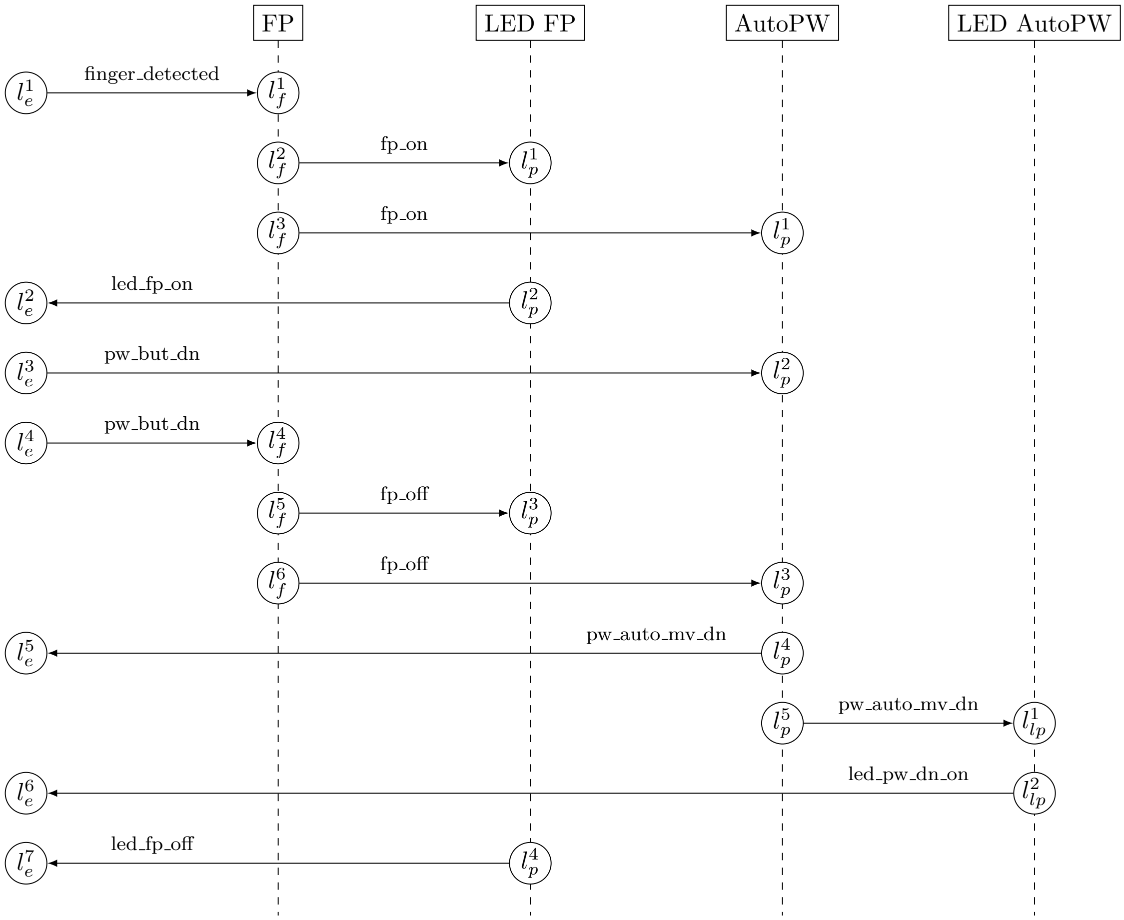

Figure 5.79.: Interaction Test Scenario MSC58

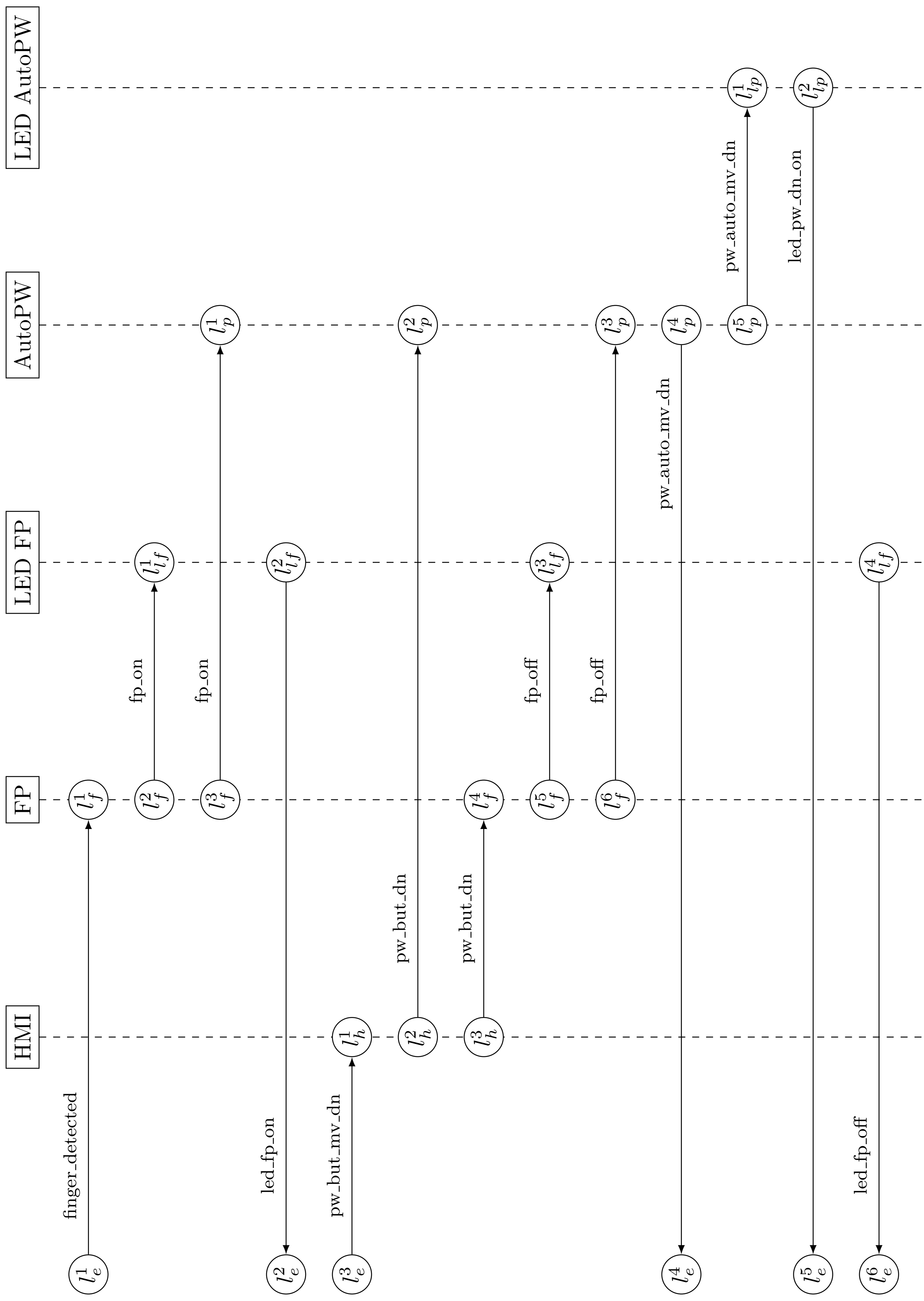

Figure 5.80.: Interaction Test Scenario MSC59

Interaction Test Scenario MSC60 The interaction test scenario *MSC60* defines a scenario between the *Remote Control Key Controller* (RCK Ctrl), the *Central Locking System* (CLS) and the *Alarm System* (AS). It is depicted in Fig. 5.81 and Fig. 5.82, where the dashed horizontal line represents a cut for better readability. The scenario describes the unlocking of the central locking system in combination with the disabling of the alarm monitoring of the alarm system. In addition, the scenario defines the activation of the safety function, i.e., the re-locking of the central locking system as well as the re-enabling of the alarm monitoring after the corresponding unlocking time elapsed. Furthermore, the interior alarm is triggered.

Interaction Test Scenario MSC61 The interaction test scenario *MSC61* shown in Fig. 5.83, defines a scenario between the *Central Locking System* (CLS), the *Automatic Power Window* (AutoPW) and the *LED Automatic Power Window* (LED AutoPW). The scenario describes the automated upwards movement of the automatic power window if the central locking system gets activated and the stopping of the window movement if the window reaches its upper position. In addition, the scenario defines the turning on/off of the corresponding LEDs.

Interaction Test Scenario MSC62 The interaction test scenario *MSC62* defines a scenario between the *Human Machine Interface* (HMI), the *Exterior Mirror* (EM), the *LED Exterior Mirror Bottom* (LED EMB), the *LED Exterior Mirror Top* (LED EMT), the *LED Exterior Mirror Right* (LED EMR) and the *LED Exterior Mirror Left* (LED EML). It is depicted in Fig. 5.84, Fig. 5.85 and Fig. 5.86, where the dashed horizontal lines represents a cut for better readability. The scenario describes the movement of the exterior mirror as well as the turning on/off of the corresponding LEDs.

Interaction Test Scenario MSC63 The interaction test scenario *MSC63* shown in Fig. 5.87, defines a scenario between the *Exterior Mirror with Heating* (EMH) and the *LED Exterior Mirror Heating* (LED EMH). The scenario describes the activation of the mirror heater if the outside temperature is too low as well as its deactivation, based on the elapsed heating time. In addition, the scenario defines the turning on/off of the corresponding LED.

Interaction Test Scenario MSC64 The interaction test scenario *MSC64* depicted in Fig. 5.88, defines a scenario between the *Human Machine Interface* (HMI) and the *Alarm System* (AS). The scenario describes the activation of the alarm system as well as the enabling of the alarm monitoring, based on the locking of the car. In addition, the scenario defines the triggering of the alarm and the sending of the silent alarm after the alarm time elapsed. The scenario ends with the disabling of the alarm monitoring and the deactivation of the alarm system. In contrast to *MSC22*, the interior alarm is stopped as well.

Interaction Test Scenario MSC65 The interaction test scenario *MSC65* shown in Fig. 5.89, describes a scenario between the *Environment* (no identifier) and the *Central Locking System* (CLS). The scenario specifies the locking of the car due to the information, that the car drives with a certain speed (car_drives). The scenario ends with the unlocking of the car by open the car door. Only environment and CLS are involved, as no other components are influenced by these signals.

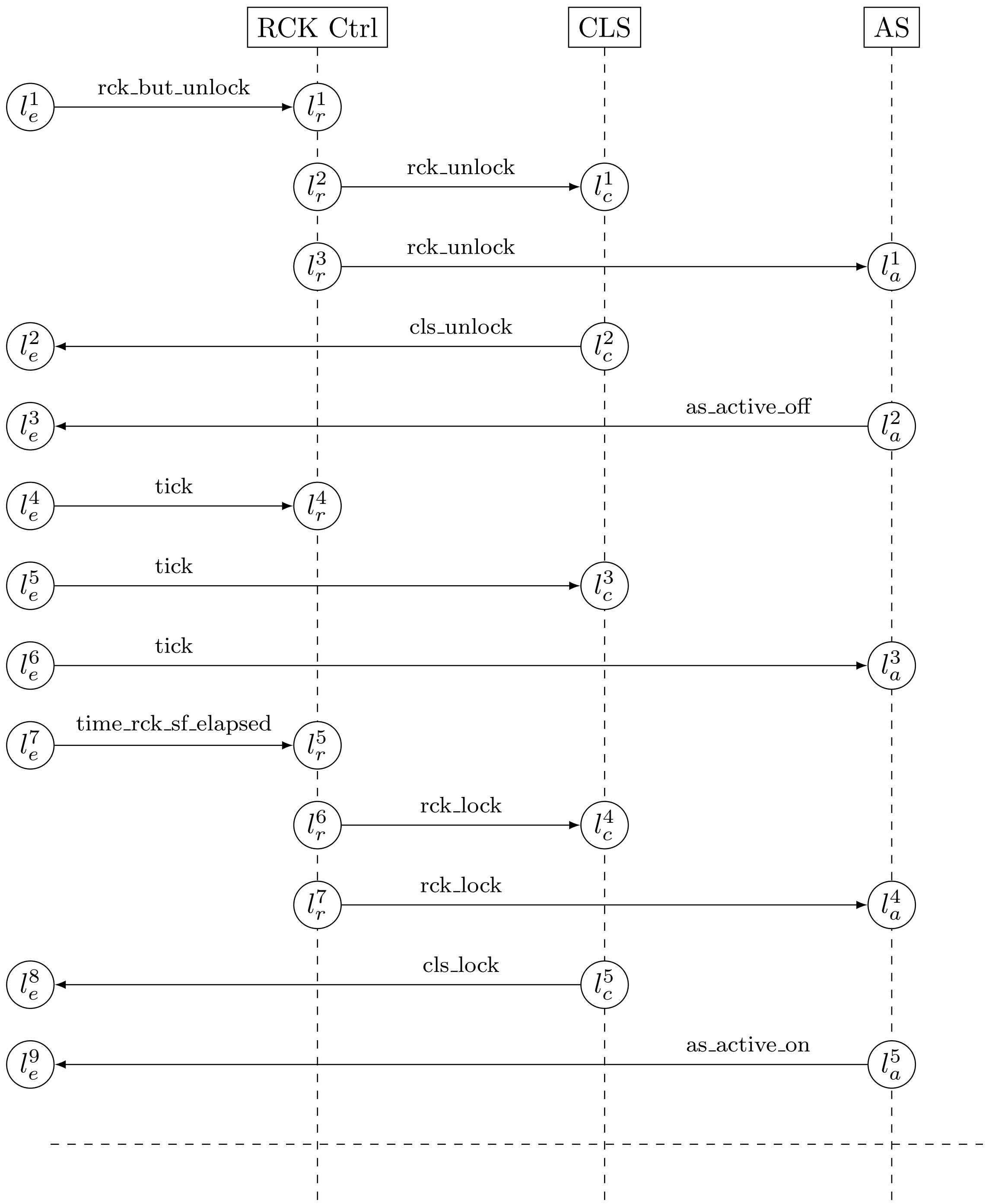

Figure 5.81.: Interaction Test Scenario MSC60 (1)

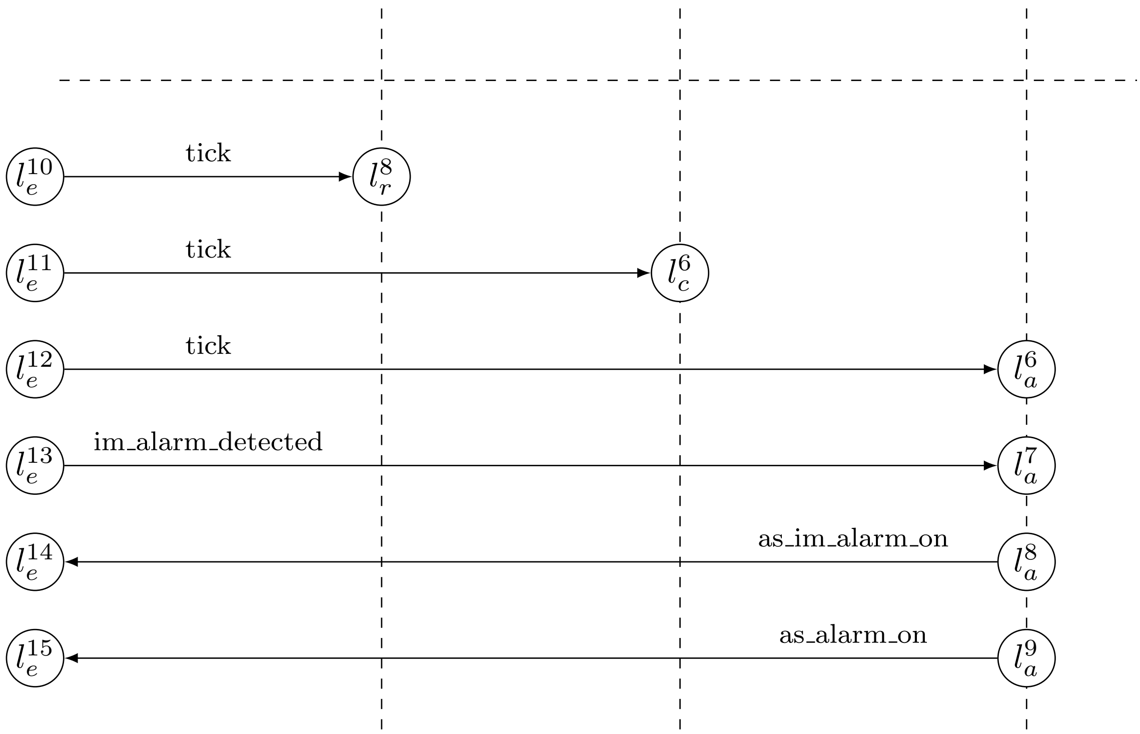

Figure 5.82.: Interaction Test Scenario MSC60 (2)

Based on the specifications of the integration test scenarios, we describe the mapping of each scenario to its corresponding product variants as well as the definition of the integration test models in the following section.

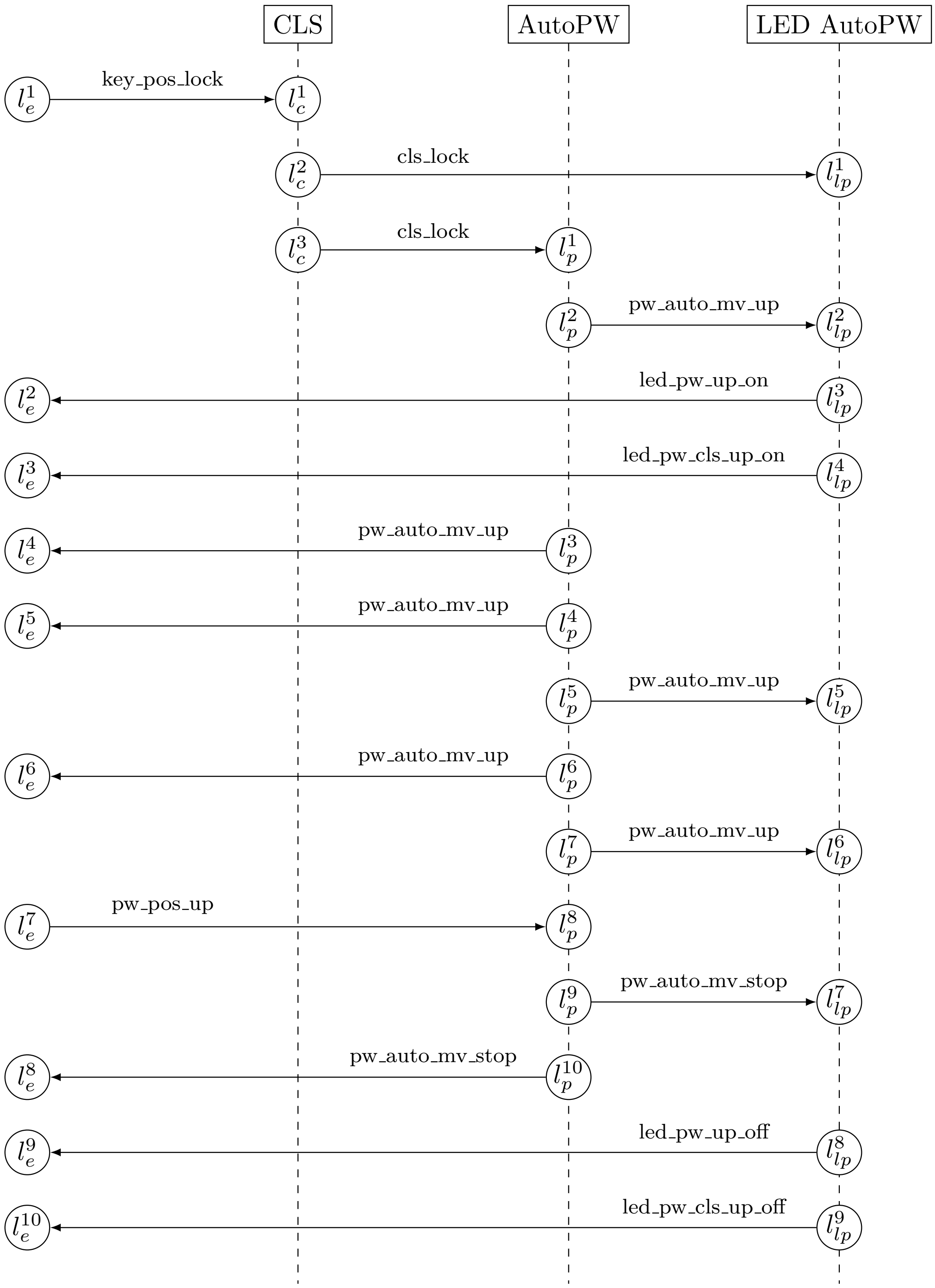

Figure 5.83.: Interaction Test Scenario MSC61

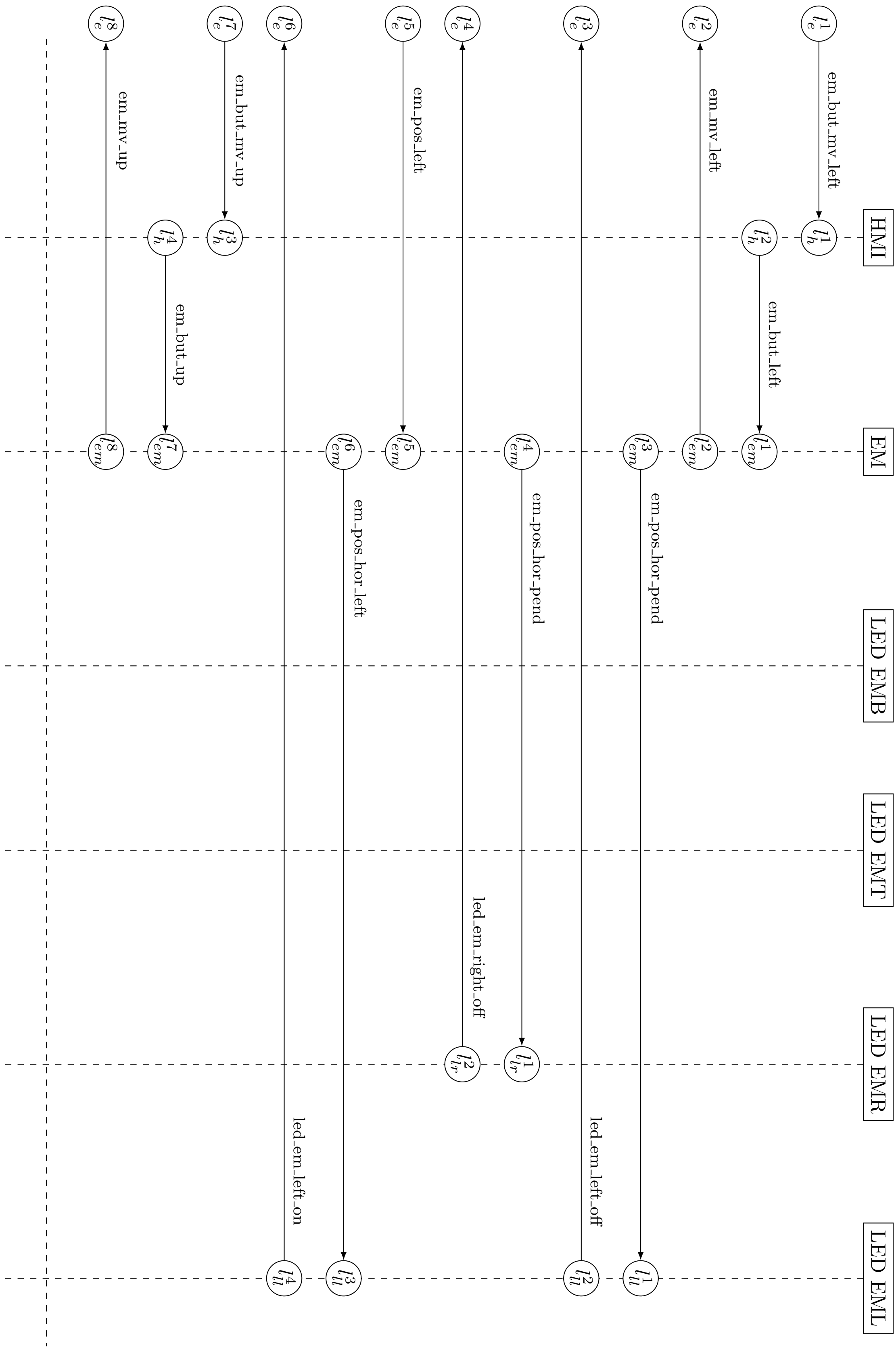

Figure 5.84.: Interaction Test Scenario MSC62 (1)

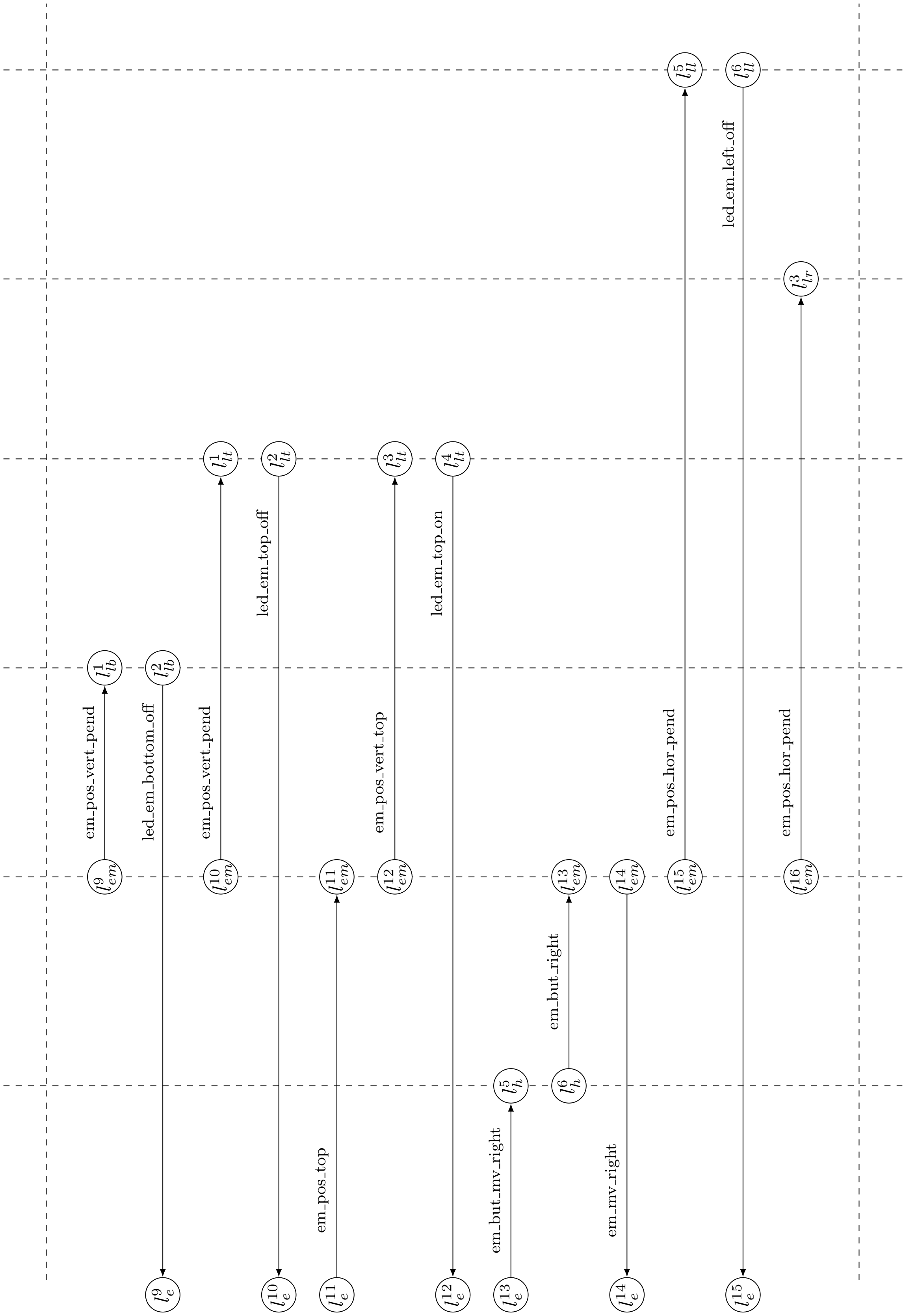

Figure 5.85.: Interaction Test Scenario MSC62 (2)

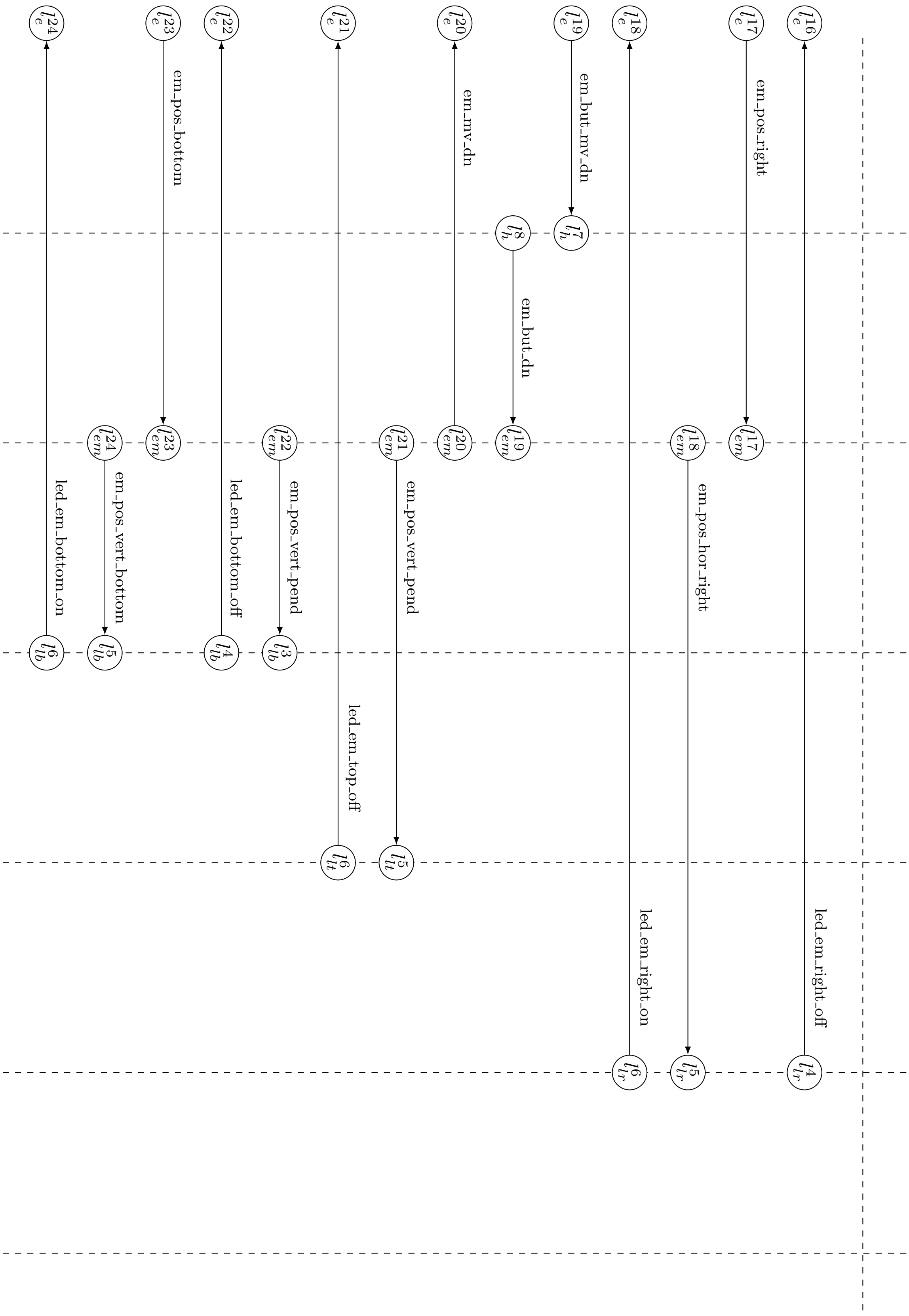

Figure 5.86.: Interaction Test Scenario MSC62 (3)

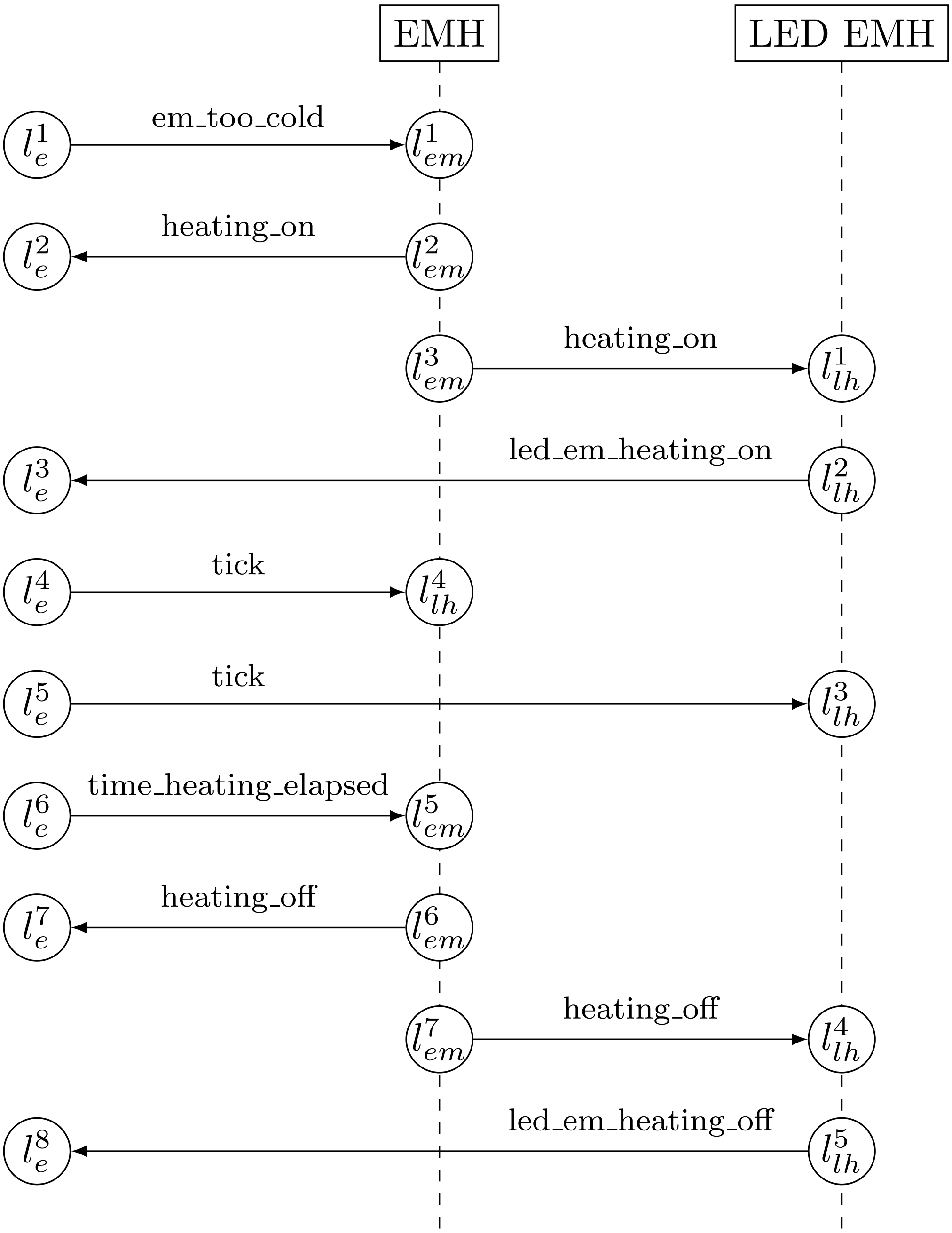

Figure 5.87.: Interaction Test Scenario MSC63

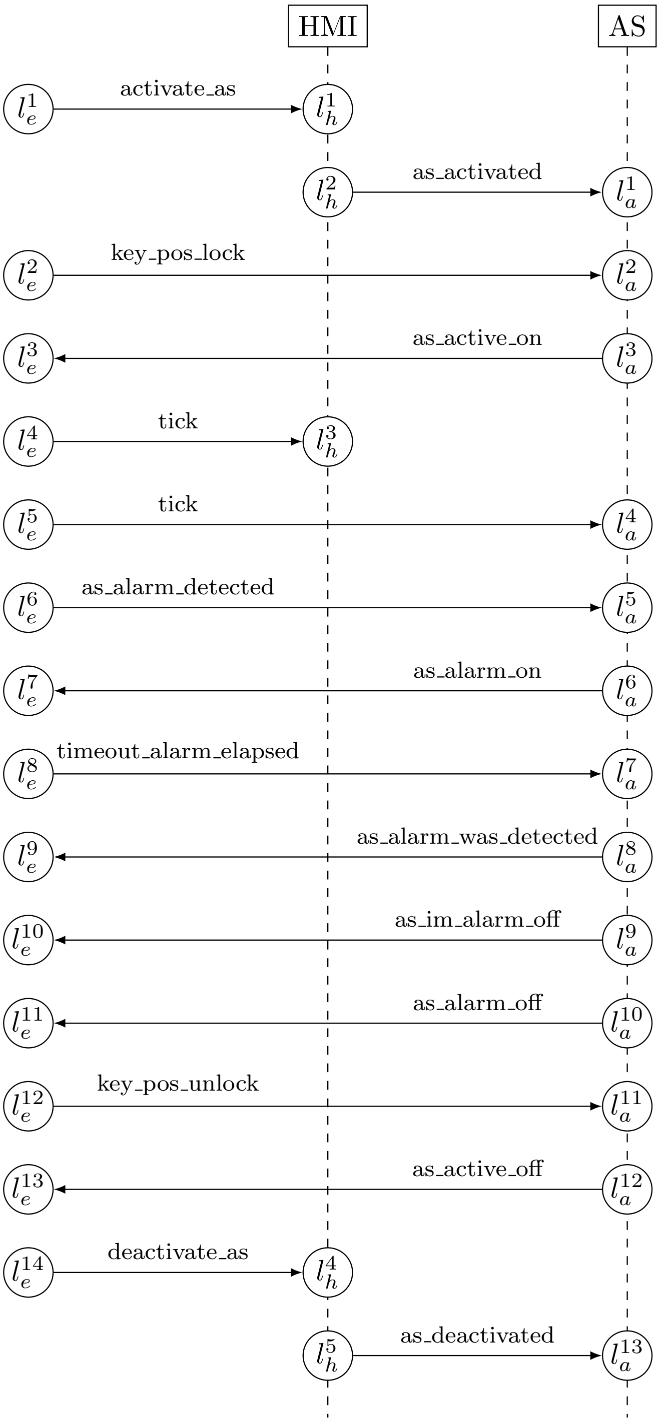

Figure 5.88.: Interaction Test Scenario MSC64

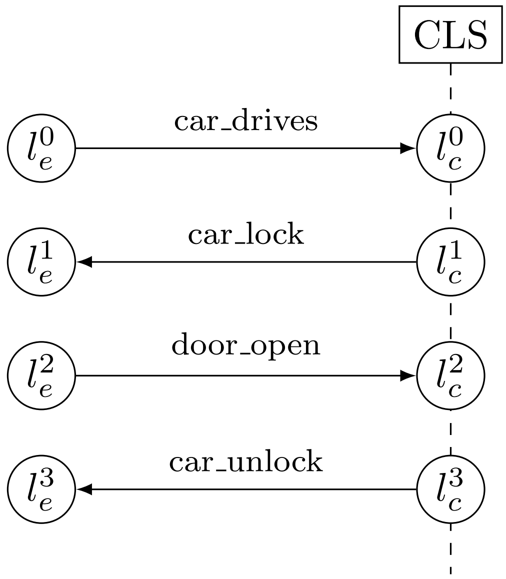

Figure 5.89.: Interaction Test Scenario MSC65

5.2. Integration Test Model Variants

In this section, we map each interaction test scenario to the product variants it is valid for as shown in Tab. 5.1, Tab. 5.2 and Tab. 5.3. Thus, the set of message sequence charts, i.e., the integration test model for each product variant is directly deducible from the provided mapping. Furthermore, we are able to deduce the integration test model deltas specified as the set of additions and removals of test scenarios by comparing the corresponding columns with the column of the core product $P0$.

Based on the specification of the integration test models as well as the specification of the architecture model variants and component state machine test model variants, we describe the aggregation of those models to the architecture test model variants for the representative subset of product variants in the next section.

	P0	*P1*	*P2*	*P3*	*P4*	*P5*	*P6*	*P7*	*P8*	*P9*	*P10*	*P11*	*P12*	*P13*	*P14*	*P15*	*P16*	*P17*
MSC1	X			X		X	X				X	X		X			X	X
MSC2	X			X		X	X				X	X		X			X	X
MSC3	X			X		X	X				X	X		X			X	X
MSC4	X		X		X			X			X	X						
MSC5	X			X		X	X				X	X		X			X	X
MSC6	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
MSC7		X							X	X	X	X		X			X	
MSC8		X	X							X			X		X	X		
MSC9		X	X					X	X				X		X	X		
MSC10		X	X							X			X		X	X		
MSC11		X	X		X					X			X		X	X		
MSC12		X				X	X		X	X	X	X		X			X	
MSC13		X	X		X					X			X		X	X		
MSC14		X	X		X			X	X	X	X				X	X		
MSC15		X	X	X		X	X		X	X			X	X	X	X	X	X
MSC16		X	X		X					X			X		X	X		
MSC17		X	X	X	X					X		X	X	X	X	X	X	
MSC18		X	X		X					X			X		X	X		
MSC19		X	X	X	X	X	X			X		X			X	X	X	
MSC20		X				X	X	X		X		X			X	X	X	
MSC21		X	X	X	X	X	X			X		X			X	X	X	

Table 5.1.: Mapping of Interaction Test Scenarios MSC1-MSC21 to Product Variants

	P0	*P1*	*P2*	*P3*	*P4*	*P5*	*P6*	*P7*	*P8*	*P9*	*P10*	*P11*	*P12*	*P13*	*P14*	*P15*	*P16*	*P17*
MSC22			X	X														
MSC23		X							X	X								
MSC24		X	X		X			X	X	X			X		X	X		
MSC25		X	X	X						X		X			X	X	X	
MSC26		X								X		X			X	X	X	
MSC27		X	X	X						X		X			X	X	X	
MSC28			X		X					X			X		X	X		
MSC29			X		X					X			X		X	X		
MSC30			X		X					X			X		X	X		
MSC31			X		X					X		X	X	X	X		X	
MSC32			X							X		X			X		X	
MSC33			X		X					X		X			X		X	
MSC34			X		X					X		X	X	X	X	X	X	
MSC35				X		X	X		X	X				X			X	
MSC36				X		X	X					X						
MSC37				X		X	X			X		X					X	
MSC38				X			X			X		X		X			X	
MSC39				X			X					X		X			X	
MSC40				X			X					X		X			X	
MSC41				X			X					X		X			X	
MSC42				X			X					X		X			X	

Table 5.2.: Mapping of Interaction Test Scenarios MSC22-MSC42 to Product Variants

	P0	*P1*	*P2*	*P3*	*P4*	*P5*	*P6*	*P7*	*P8*	*P9*	*P10*	*P11*	*P12*	*P13*	*P14*	*P15*	*P16*	*P17*
MSC43				X			X					X		X			X	
MSC44				X								X		X			X	
MSC45				X						X		X		X			X	
MSC46				X		X	X					X						
MSC47				X		X											X	
MSC48				X		X	X					X						
MSC49						X	X					X						
MSC50						X	X					X						
MSC51						X	X			X		X					X	
MSC52						X	X			X		X					X	
MSC53						X	X				X	X		X			X	
MSC54									X	X								
MSC55									X	X								
MSC56									X	X								
MSC57									X	X								
MSC58									X	X								
MSC59									X	X								
MSC60										X		X			X	X		
MSC61										X								
MSC62											X							
MSC63														X				
MSC64		X			X	X	X			X		X			X	X	X	

Table 5.3.: Mapping of Interaction Test Scenarios MSC43-MSC64 to Product Variants

6 BCS Architecture Test Model

In this section, we integrate the previously defined test models for the definition of the delta-oriented architecture test models of the BCS SPL case study. An architecture test model comprises an architecture model, the corresponding set of component state machine test models as well as an integration test model. The BCS architecture test models for the representative subset of product variants as well as the corresponding architecture test model deltas are specified as follows.

Core Architecture Test Model P0 The architecture test model $P0$ is used as the core architecture test model for the core product $P0$. Based on its feature configuration (cf. Tab. 2.1), the test model is assembled as follows.

- The corresponding core architecture model $P0$ is specified in Sect. 3.4.
- The core set of state machine test models contains four test models (cf. Sect. 4.1, Sect. 4.3) such that for every component defined in the core architecture model there is one state machine test model.
- The corresponding core integration test model for $P0$ is specified in Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

Due to the selection as core model, we do not specify an architecture test model delta.

Architecture Test Model P1 The architecture test model variant $P1$ corresponds to product $P1$. Based on its feature configuration (cf. Tab. 2.1), the test model is assembled as follows.

- The corresponding architecture model $P1$ is specified in Sect. 3.4.
- The set of state machine test models contains eight test models (cf. Sect. 4.1, Sect. 4.3) such that for every component defined in the architecture model there is one state machine test model.
- The corresponding integration test model for $P1$ is specified in Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

For the transformation of the core architecture model, an architecture test model delta is defined comprising

- a set of eight architecture model deltas as defined in Sect. 3.3 in Tab. 3.4,

- a set of five state machine test model deltas as defined in Sect. 4.2, and
- a set of 26 integration test model deltas deducible from the Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

Architecture Test Model P2 The architecture test model variant *P*2 corresponds to product *P*2. Based on its feature configuration (cf. Tab. 2.1), the test model is assembled as follows.

- The corresponding architecture model *P*2 is specified in Sect. 3.4.
- The set of state machine test models contains seven test models (cf. Sect. 4.1, Sect. 4.3) such that for every component defined in the architecture model there is one state machine test model.
- The corresponding integration test model for *P*2 is specified in Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

For the transformation of the core architecture model, an architecture test model delta is defined comprising

- a set of seven architecture model deltas as defined in Sect. 3.3 in Tab. 3.4,
- a set of eight state machine test model deltas as defined in Sect. 4.2, and
- a set of 27 integration test model deltas deducible from the Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

Architecture Test Model P3 The architecture test model variant *P*3 corresponds to product *P*3. Based on its feature configuration (cf. Tab. 2.1), the test model is assembled as follows.

- The corresponding architecture model *P*3 is specified in Sect. 3.4.
- The set of state machine test models contains 17 test models (cf. Sect. 4.1, Sect. 4.3) such that for every component defined in the architecture model there is one state machine test model.
- The corresponding integration test model for *P*3 is specified in Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

For the transformation of the core architecture model, an architecture test model delta is defined comprising

- a set of 10 architecture model deltas as defined in Sect. 3.3 in Tab. 3.4,
- a set of eight state machine test model deltas as defined in Sect. 4.2, and
- a set of 22 integration test model deltas deducible from the Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

Architecture Test Model P4 The architecture test model variant *P*4 corresponds to product *P*4. Based on its feature configuration (cf. Tab. 2.1), the test model is assembled as follows.

- The corresponding architecture model *P*4 is specified in Sect. 3.4.
- The set of state machine test models contains seven test models (cf. Sect. 4.1, Sect. 4.3) such that for every component defined in the architecture model there is one state machine test model.
- The corresponding integration test model for *P*4 is specified in Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

For the transformation of the core architecture model, an architecture test model delta is defined comprising

- a set of eight architecture model deltas as defined in Sect. 3.3 in Tab. 3.4,
- a set of seven state machine test model deltas as defined in Sect. 4.2, and
- a set of 20 integration test model deltas deducible from the Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

Architecture Test Model P5 The architecture test model variant *P*5 corresponds to product *P*5. Based on its feature configuration (cf. Tab. 2.1), the test model is assembled as follows.

- The corresponding architecture model *P*5 is specified in Sect. 3.4.
- The set of state machine test models contains 16 test models (cf. Sect. 4.1, Sect. 4.3) such that for every component defined in the architecture model there is one state machine test model.
- The corresponding integration test model for *P*5 is specified in Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

For the transformation of the core architecture model, an architecture test model delta is defined comprising

- a set of nine architecture model deltas as defined in Sect. 3.3 in Tab. 3.4,
- a set of five state machine test model deltas as defined in Sect. 4.2, and
- a set of 18 integration test model deltas deducible from the Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

Architecture Test Model P6 The architecture test model variant *P*6 corresponds to product *P*6. Based on its feature configuration (cf. Tab. 2.1), the test model is assembled as follows.

- The corresponding architecture model *P*6 is specified in Sect. 3.4.

- The set of state machine test models contains 18 test models (cf. Sect. 4.1, Sect. 4.3) such that for every component defined in the architecture model there is one state machine test model.
- The corresponding integration test model for *P*6 is specified in Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

For the transformation of the core architecture model, an architecture test model delta is defined comprising

- a set of 12 architecture model deltas as defined in Sect. 3.3 in Tab. 3.4,
- a set of six state machine test model deltas as defined in Sect. 4.2, and
- a set of 23 integration test model deltas deducible from the Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

Architecture Test Model P7 The architecture test model variant *P*7 corresponds to product *P*7. Based on its feature configuration (cf. Tab. 2.1), the test model is assembled as follows.

- The corresponding architecture model *P*7 is specified in Sect. 3.4.
- The set of state machine test models contains four test models (cf. Sect. 4.1, Sect. 4.3) such that for every component defined in the architecture model there is one state machine test model.
- The corresponding integration test model for *P*7 is specified in Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

For the transformation of the core architecture model, an architecture test model delta is defined comprising

- a set of one architecture model delta as defined in Sect. 3.3 in Tab. 3.4,
- an empty set of state machine test model deltas as defined in Sect. 4.2, and
- a set of eight integration test model deltas deducible from the Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

Architecture Test Model P8 The architecture test model variant *P*8 corresponds to product *P*8. Based on its feature configuration (cf. Tab. 2.1), the test model is assembled as follows.

- The corresponding architecture model *P*8 is specified in Sect. 3.4.
- The set of state machine test models contains 11 test models (cf. Sect. 4.1, Sect. 4.3) such that for every component defined in the architecture model there is one state machine test model.
- The corresponding integration test model for *P*8 is specified in Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

For the transformation of the core architecture model, an architecture test model delta is defined comprising

- a set of six architecture model deltas as defined in Sect. 3.3 in Tab. 3.4,
- a set of two state machine test model deltas as defined in Sect. 4.2, and
- a set of 19 integration test model deltas deducible from the Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

Architecture Test Model P9 The architecture test model variant $P9$ corresponds to product $P9$. Based on its feature configuration (cf. Tab. 2.2), the test model is assembled as follows.

- The corresponding architecture model $P9$ is specified in Sect. 3.4.
- The set of state machine test models contains 19 test models (cf. Sect. 4.1, Sect. 4.3) such that for every component defined in the architecture model there is one state machine test model.
- The corresponding integration test model for $P9$ is specified in Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

For the transformation of the core architecture model, an architecture test model delta is defined comprising

- a set of 16 architecture model deltas as defined in Sect. 3.3 in Tab. 3.5,
- a set of 11 state machine test model deltas as defined in Sect. 4.2, and
- a set of 46 integration test model deltas deducible from the Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

Architecture Test Model P10 The architecture test model variant $P10$ corresponds to product $P10$. Based on its feature configuration (cf. Tab. 2.2), the test model is assembled as follows.

- The corresponding architecture model $P10$ is specified in Sect. 3.4.
- The set of state machine test models contains nine test models (cf. Sect. 4.1, Sect. 4.3) such that for every component defined in the architecture model there is one state machine test model.
- The corresponding integration test model for $P10$ is specified in Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

For the transformation of the core architecture model, an architecture test model delta is defined comprising

- a set of two architecture model deltas as defined in Sect. 3.3 in Tab. 3.5,

- a set of one state machine test model delta as defined in Sect. 4.2, and
- a set of five integration test model deltas deducible from the Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

Architecture Test Model P11 The architecture test model variant $P11$ corresponds to product $P11$. Based on its feature configuration (cf. Tab. 2.2), the test model is assembled as follows.

- The corresponding architecture model $P11$ is specified in Sect. 3.4.
- The set of state machine test models contains 14 test models (cf. Sect. 4.1, Sect. 4.3) such that for every component defined in the architecture model there is one state machine test model.
- The corresponding integration test model for $P11$ is specified in Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

For the transformation of the core architecture model, an architecture test model delta is defined comprising

- a set of 12 architecture model deltas as defined in Sect. 3.3 in Tab. 3.5,
- a set of seven state machine test model deltas as defined in Sect. 4.2, and
- a set of 32 integration test model deltas deducible from the Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

Architecture Test Model P12 The architecture test model variant $P12$ corresponds to product $P12$. Based on its feature configuration (cf. Tab. 2.2), the test model is assembled as follows.

- The corresponding architecture model $P12$ is specified in Sect. 3.4.
- The set of state machine test models contains seven test models (cf. Sect. 4.1, Sect. 4.3) such that for every component defined in the architecture model there is one state machine test model.
- The corresponding integration test model for $P12$ is specified in Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

For the transformation of the core architecture model, an architecture test model delta is defined comprising

- a set of seven architecture model deltas as defined in Sect. 3.3 in Tab. 3.5,
- a set of seven state machine test model deltas as defined in Sect. 4.2, and
- a set of 20 integration test model deltas deducible from the Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

Architecture Test Model P13 The architecture test model variant $P13$ corresponds to product $P13$. Based on its feature configuration (cf. Tab. 2.2), the test model is assembled as follows.

- The corresponding architecture model $P13$ is specified in Sect. 3.4.
- The set of state machine test models contains nine test models (cf. Sect. 4.1, Sect. 4.3) such that for every component defined in the architecture model there is one state machine test model.
- The corresponding integration test model for $P13$ is specified in Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

For the transformation of the core architecture model, an architecture test model delta is defined comprising

- a set of seven architecture model deltas as defined in Sect. 3.3 in Tab. 3.5,
- a set of four state machine test model deltas as defined in Sect. 4.2, and
- a set of 18 integration test model deltas deducible from the Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

Architecture Test Model P14 The architecture test model variant $P14$ corresponds to product $P14$. Based on its feature configuration (cf. Tab. 2.2), the test model is assembled as follows.

- The corresponding architecture model $P14$ is specified in Sect. 3.4.
- The set of state machine test models contains seven test models (cf. Sect. 4.1, Sect. 4.3) such that for every component defined in the architecture model there is one state machine test model.
- The corresponding integration test model for $P14$ is specified in Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

For the transformation of the core architecture model, an architecture test model delta is defined comprising

- a set of 10 architecture model deltas as defined in Sect. 3.3 in Tab. 3.5,
- a set of 10 state machine test model deltas as defined in Sect. 4.2, and
- a set of 31 integration test model deltas deducible from the Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

Architecture Test Model P15 The architecture test model variant $P15$ corresponds to product $P15$. Based on its feature configuration (cf. Tab. 2.2), the test model is assembled as follows.

- The corresponding architecture model $P15$ is specified in Sect. 3.4.

- The set of state machine test models contains seven test models (cf. Sect. 4.1, Sect. 4.3) such that for every component defined in the architecture model there is one state machine test model.
- The corresponding integration test model for $P15$ is specified in Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

For the transformation of the core architecture model, an architecture test model delta is defined comprising

- a set of nine architecture model deltas as defined in Sect. 3.3 in Tab. 3.5,
- a set of eight state machine test model deltas as defined in Sect. 4.2, and
- a set of 28 integration test model deltas deducible from the Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

Architecture Test Model P16 The architecture test model variant $P16$ corresponds to product $P16$. Based on its feature configuration (cf. Tab. 2.2), the test model is assembled as follows.

- The corresponding architecture model $P16$ is specified in Sect. 3.4.
- The set of state machine test models contains 18 test models (cf. Sect. 4.1, Sect. 4.3) such that for every component defined in the architecture model there is one state machine test model.
- The corresponding integration test model for $P16$ is specified in Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

For the transformation of the core architecture model, an architecture test model delta is defined comprising

- a set of 14 architecture model deltas as defined in Sect. 3.3 in Tab. 3.5,
- a set of 10 state machine test model deltas as defined in Sect. 4.2, and
- a set of 30 integration test model deltas deducible from the Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

Architecture Test Model P17 The architecture test model variant $P17$ corresponds to product $P17$. Based on its feature configuration (cf. Tab. 2.2), the test model is assembled as follows.

- The corresponding architecture model $P17$ is specified in Sect. 3.4.
- The set of state machine test models contains four test models (cf. Sect. 4.1, Sect. 4.3) such that for every component defined in the architecture model there is one state machine test model.
- The corresponding integration test model for $P17$ is specified in Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

For the transformation of the core architecture model, an architecture test model delta is defined comprising

- a set of one architecture model delta as defined in Sect. 3.3 in Tab. 3.5,
- a set of one state machine test model delta as defined in Sect. 4.2, and
- a set of two integration test model deltas deducible from the Tab. 5.1, Tab. 5.2 and Tab. 5.3 in Sect. 5.2.

7 Conclusions

In this technical report, we gave an extensive overview of the test models developed for the automotive SPL case study *Body Comfort System* (BCS). We described the underlying feature-model in detail as well as the examined product variants on the component testing and integration testing levels. We used the case study to evaluate delta-oriented testing strategies for software product lines. These strategies aim for a reduction of the testing effort by reusing test artifacts.

In our model-based testing approach, we defined state machine test models for the components of the systems. They describe the components' internal behavior. These state machine models have been modified in order to represent the variability of the components. Therefore, we introduced delta-oriented state machines. For the case study, this includes the core test models, the defined delta models for the generation of test model variants and the generated component state machine test model variants of the *BCS* SPL.

For the purpose of integration testing, we used a textual representation of the software architectures for the different examined product variants of the BCS SPL. We realized the representation using a self-defined architecture description language called DeltaRx. Similar to the modified state machines, DeltaRx supports variability in form of deltas for software architectures. We defined a core product for the BCS and based upon this, architectural model variants in form of deltas needed to generate the respective products. The component behavior has been realized using the defined state machine models. In addition, a graphical representation in form of block diagrams of the product architectures has been shown. Based on the architecture models, a set of message sequence charts has been defined as variable integration test scenarios. The corresponding scenarios have been described in detail.

In further research, we are investigating the possibilities to transform our testing approach to the requirements level. This approach is based on requirements and their associated test cases described in natural language specifications. We examine if delta-oriented testing techniques are applicable to the requirements level in order to reduce the testing effort if executable system specifications are not available. This approach will be further evaluated using the *Body Comfort System* SPL case study.

Bibliography

[1] H. Cichos et al. “Model-based Coverage-Driven Test Suite Generation for Software Product Lines”. In: *MODELS’11*. 2011.

[2] P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. Addison Wesley Longman, 2001.

[3] K. Kang et al. *Feature-oriented Domain Analysis Feasibility Study*. Tech. rep. CMU/SEI-90-TR-21. Software Engineering Institute, CMU, 1990.

[4] S. Lity et al. “Delta-oriented model-based SPL regression testing”. In: *Product Line Approaches in Software Engineering (PLEASE), 2012 3rd International Workshop on*. June 2012, pp. 53–56.

[5] Malte Lochau et al. “Incremental Model-based Testing of Delta-oriented Software Product Lines”. In: *6th International Conference on Tests & Proofs*. Prague, 2012.

[6] Tobias C. Müller et al. *A Comprehensive Description of a Model-based, Continous Development Process for AUTOSAR Systems with Integrated Quality Assurance*. Informatik-Bericht. TU Braunschweig, 2009 (2009-06).

[7] Sebastian Oster et al. “Pairwise feature-interaction testing for SPLs: potentials and limitations”. In: *Proceedings of the 15th International Software Product Line Conference, Volume 2*. SPLC ’11. Munich, Germany: ACM, 2011, 6:1–6:8. DOI: 10.1145/2019136.2019143.

[8] K. Pohl, G. Böckle, and F. van der Linden. *Software Product Line Engineering - Foundations, Principles, and Techniques*. Springer, Heidelberg, 2005.

[9] Ina Schaefer. “Variability Modelling for Model-Driven Development of Software Product Lines”. In: *VaMoS*. 2010.

[10] M. Utting and B. Legeard. *Practical Model-Based Testing. A Tools Approach*. M. Kaufmann, 2007.

[11] Marius Zink. “Anwendung von MoSo-PoLiTe in einer Automotive SPL”. Masterarbeit. Technische Universität Darmstadt, June 2011.

A Appendix

A.1. Body Comfort System 150% State Machine Test Model

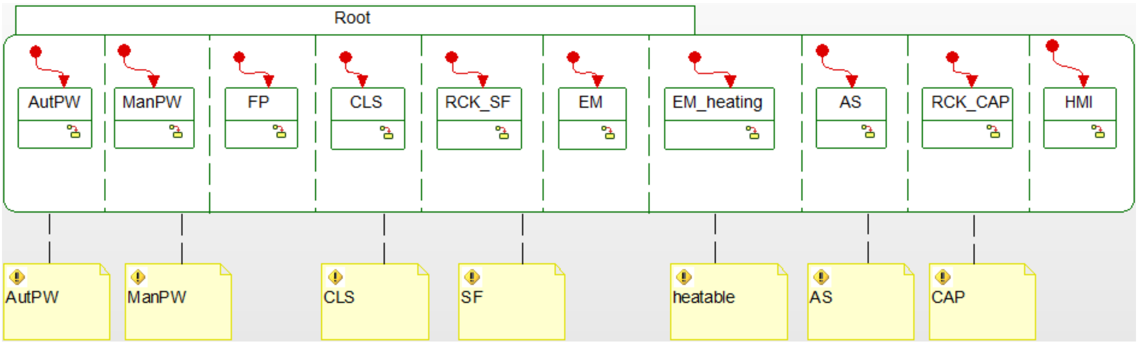

Figure A.1.: 150% State Machine BCS Root

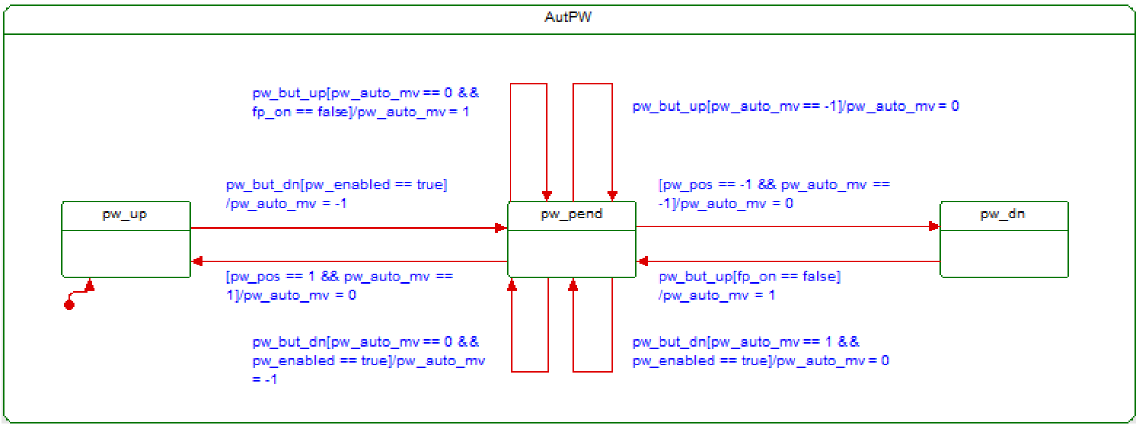

Figure A.2.: 150% Sub State Machine AutPW

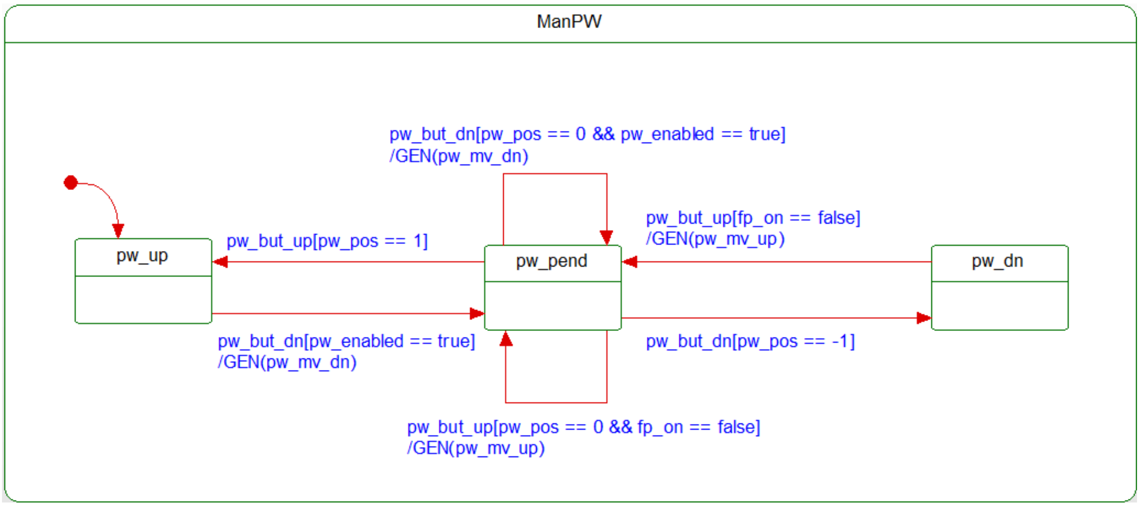

Figure A.3.: 150% Sub State Machine ManPW

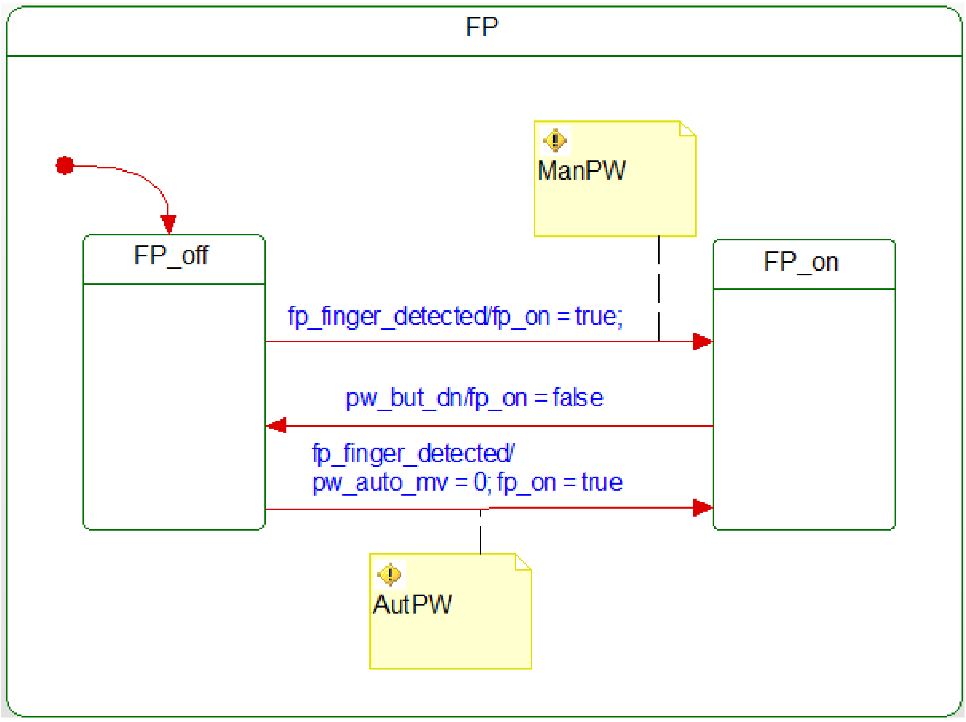

Figure A.4.: 150% Sub State Machine FP

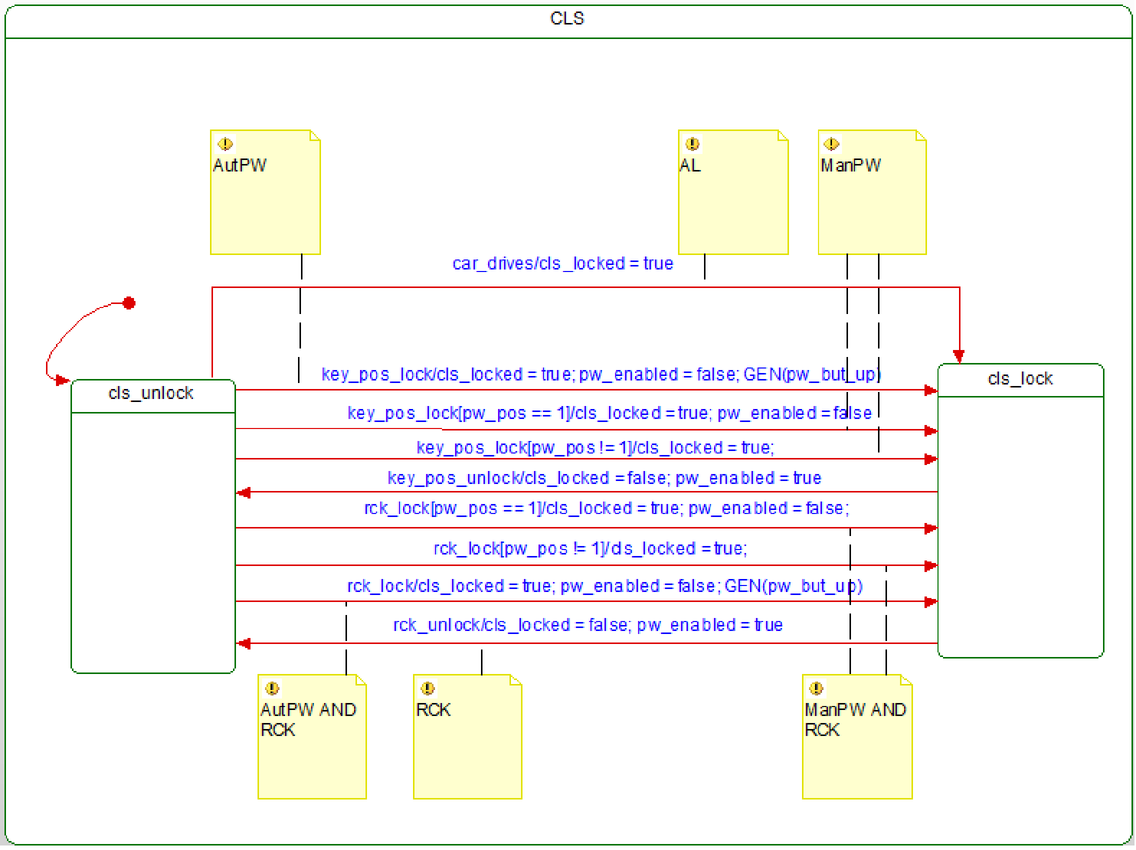

Figure A.5.: 150% Sub State Machine CLS

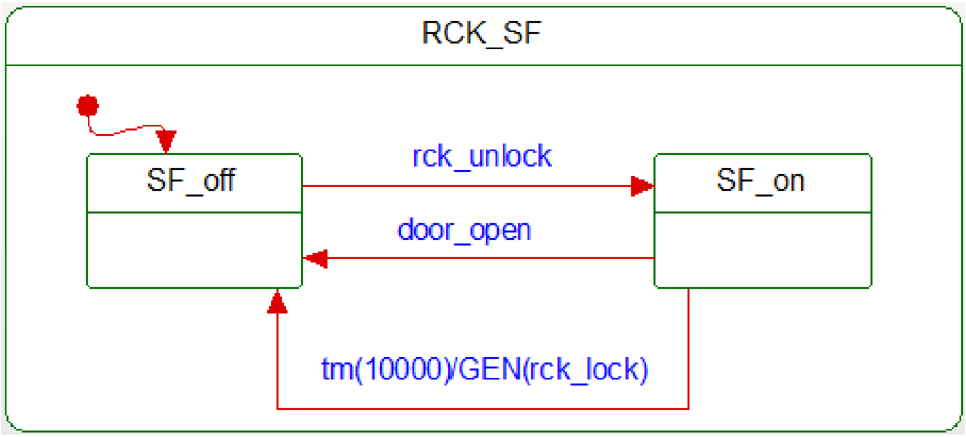

Figure A.6.: 150% Sub State Machine RCK_SF

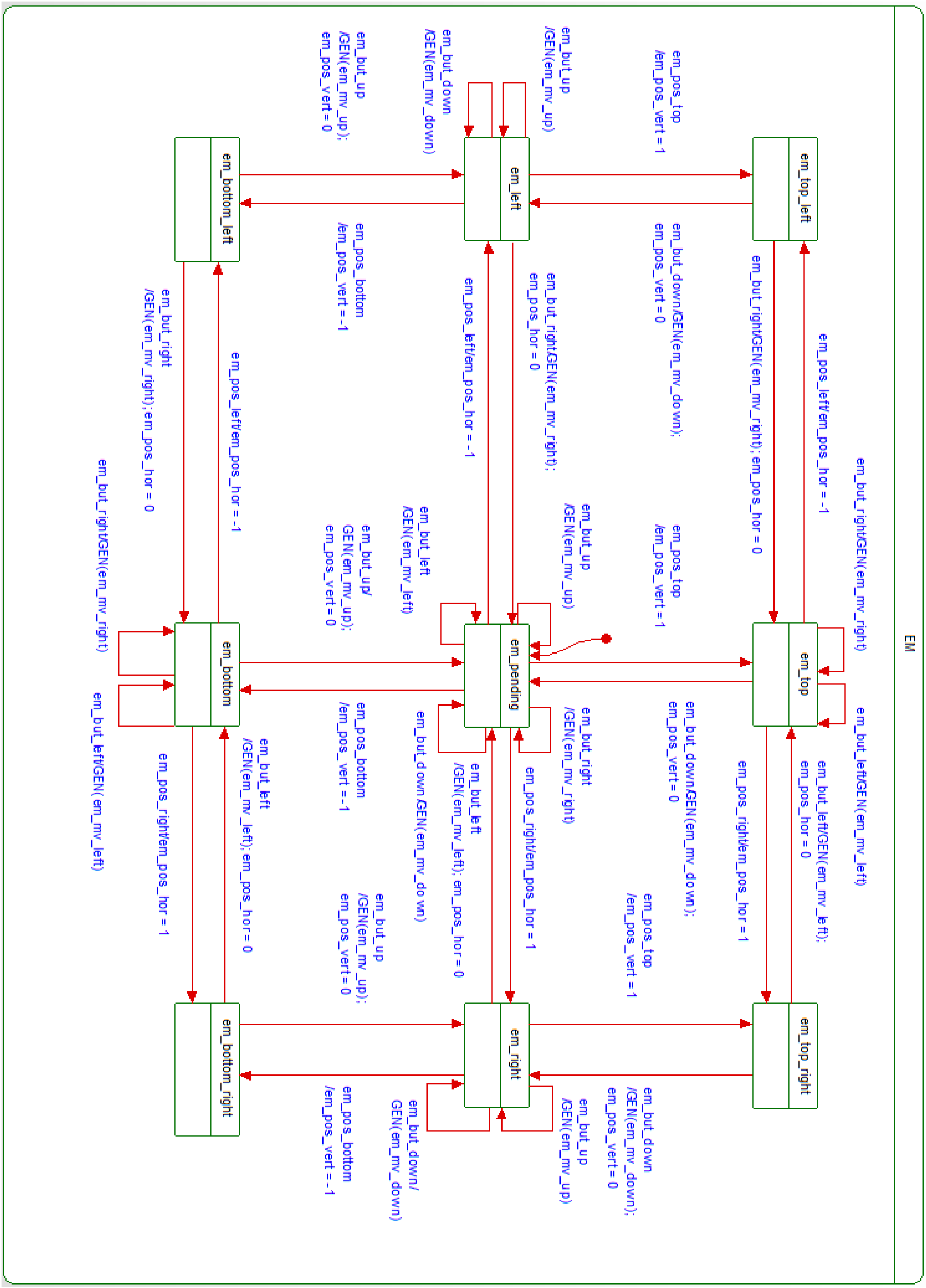

Figure A.7.: 150% Sub State Machine EM

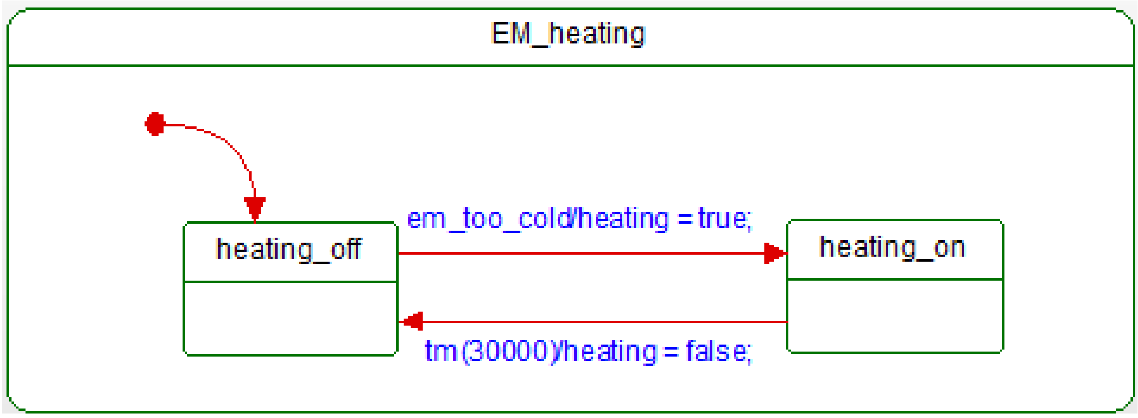

Figure A.8.: 150% Sub State Machine EM_heating

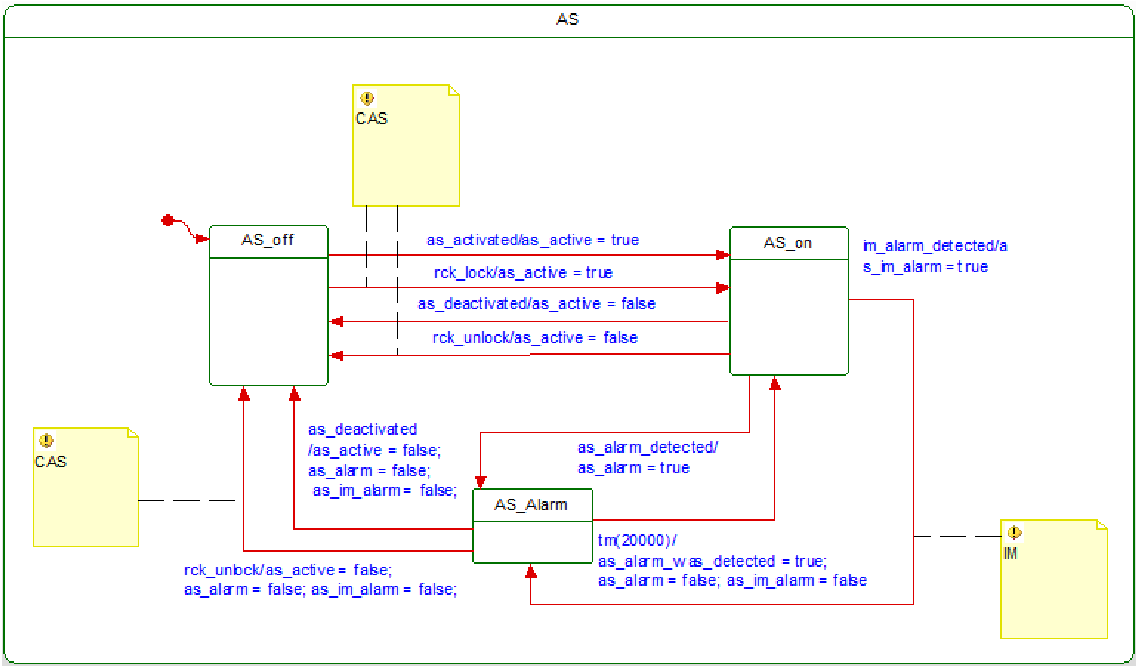

Figure A.9.: 150% Sub State Machine AS

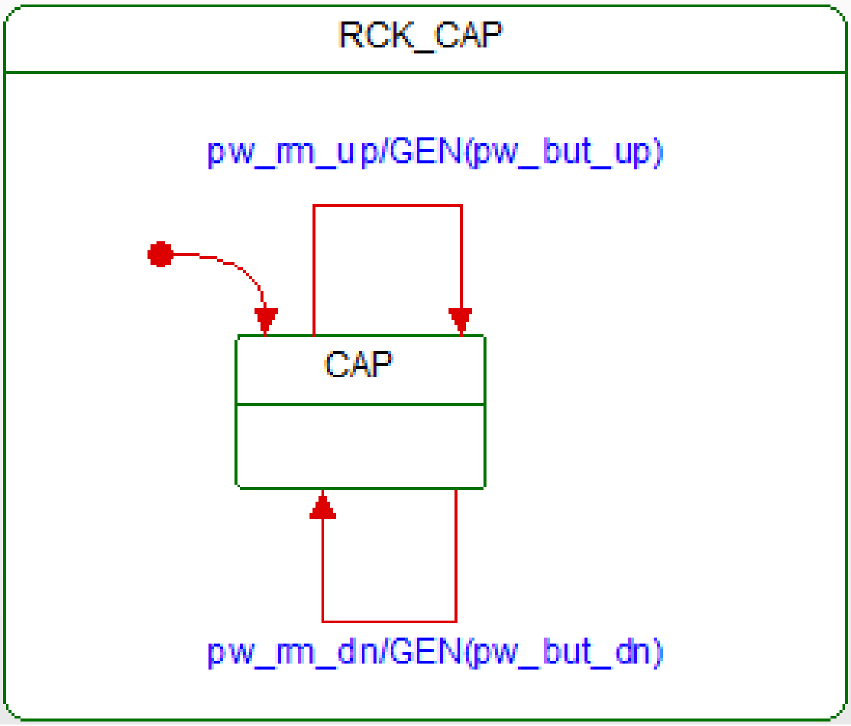

Figure A.10.: 150% Sub State Machine RCK_CAP

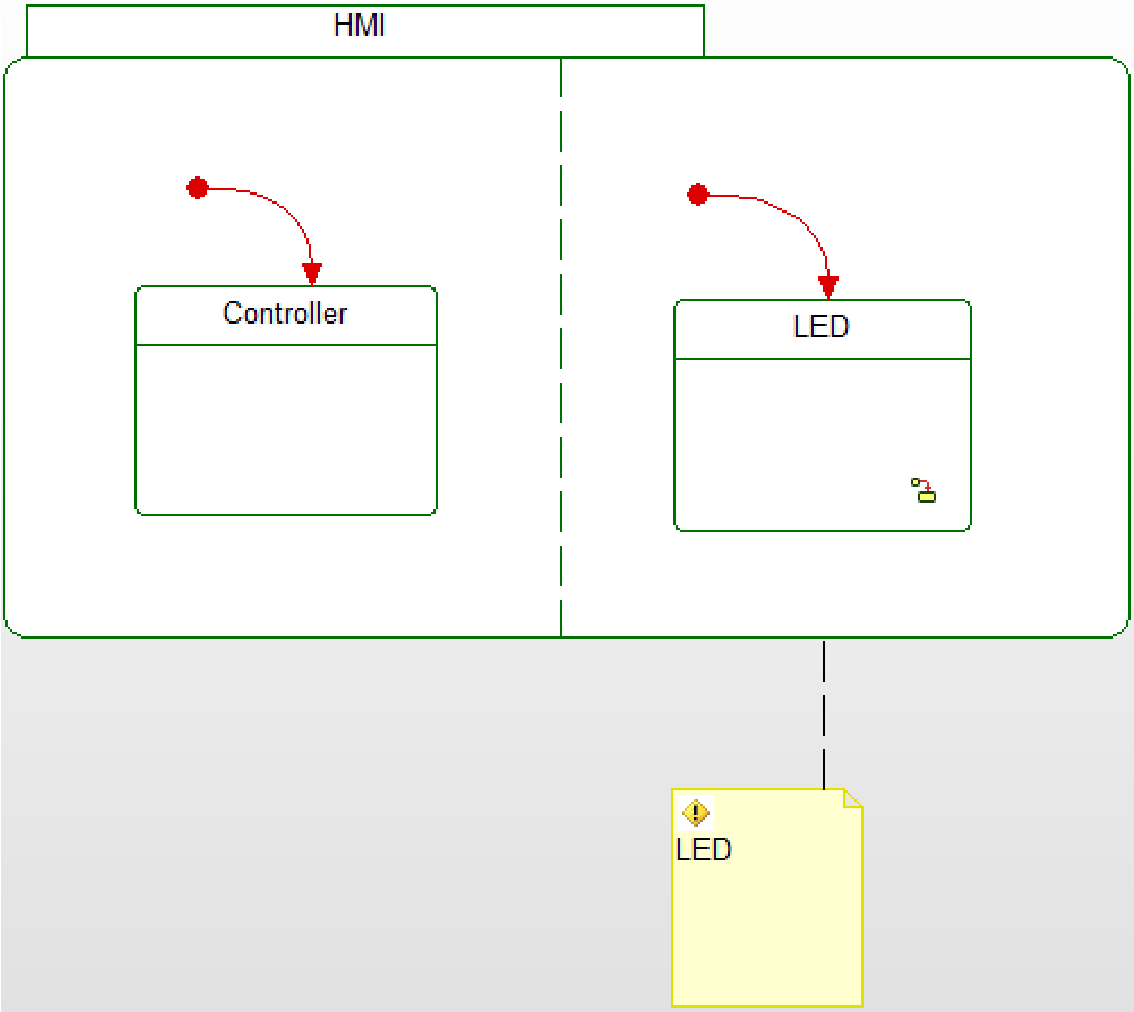

Figure A.11.: 150% Sub State Machine HMI

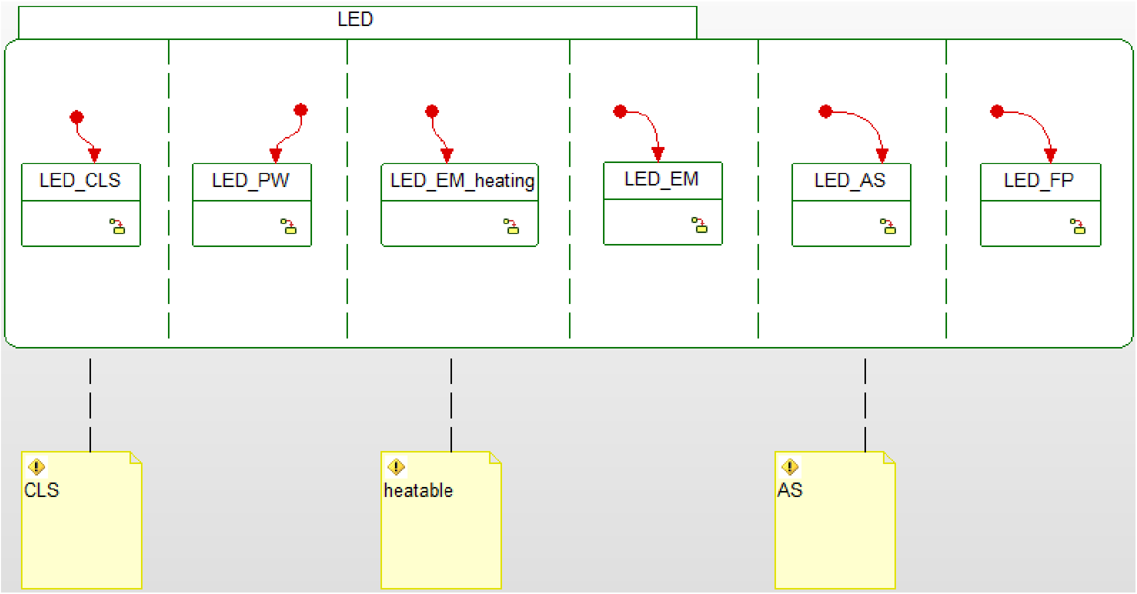

Figure A.12.: 150% Sub State Machine LED

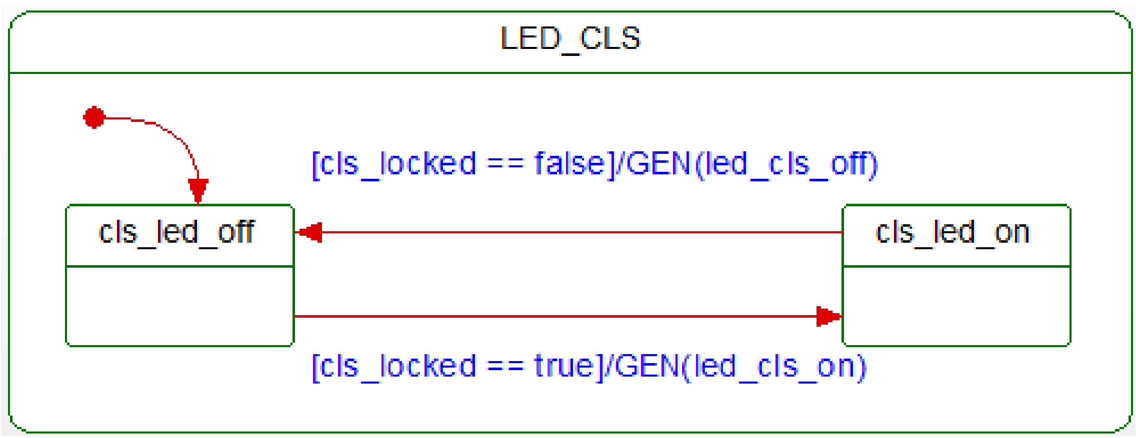

Figure A.13.: 150% Sub State Machine LED_CLS

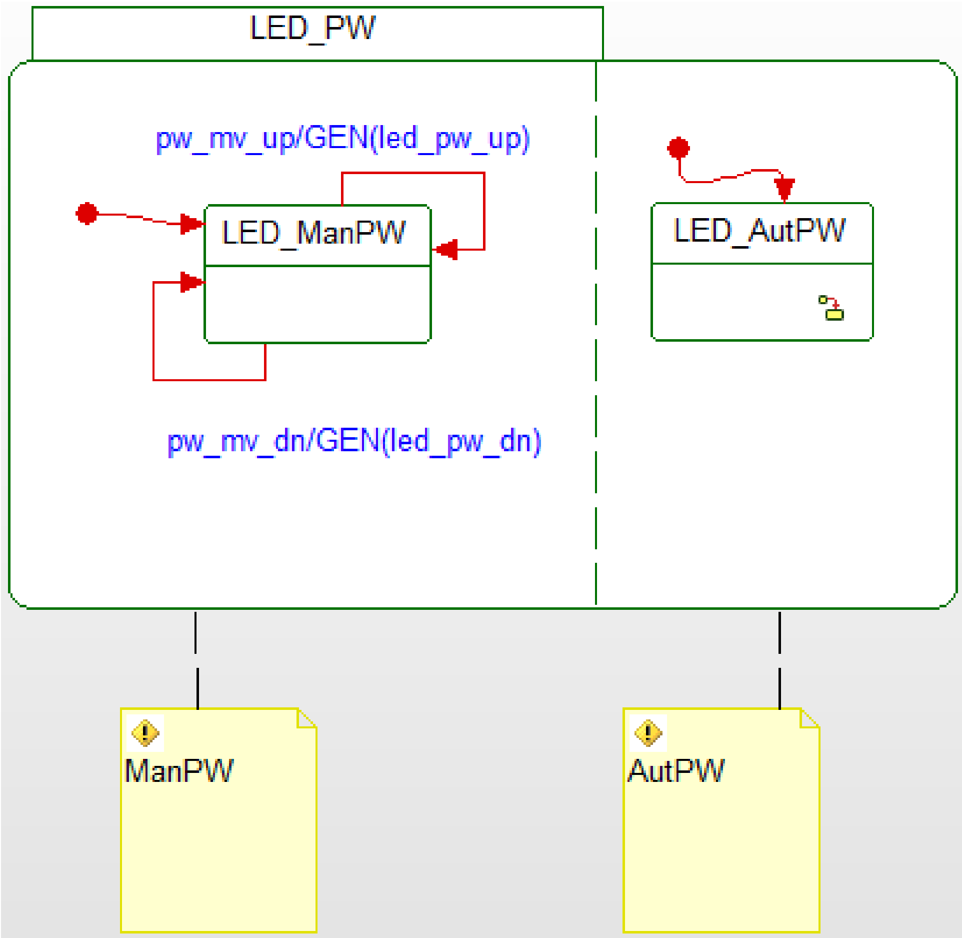

Figure A.14.: 150% Sub State Machine LED_PW

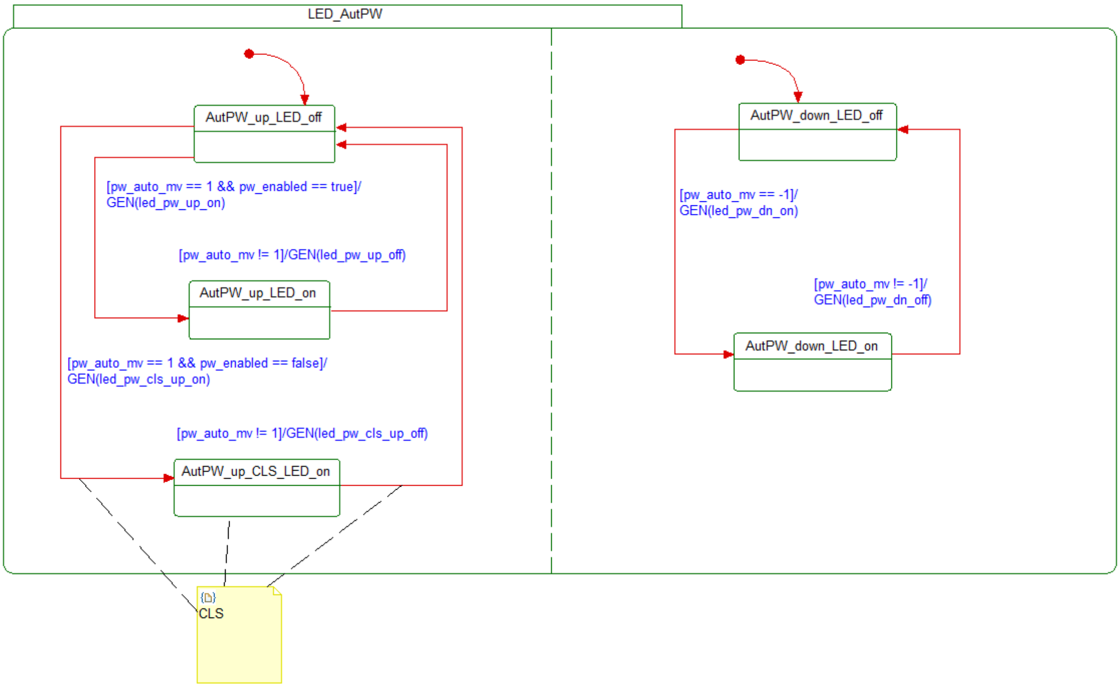

Figure A.15.: 150% Sub State Machine LED_AutPW

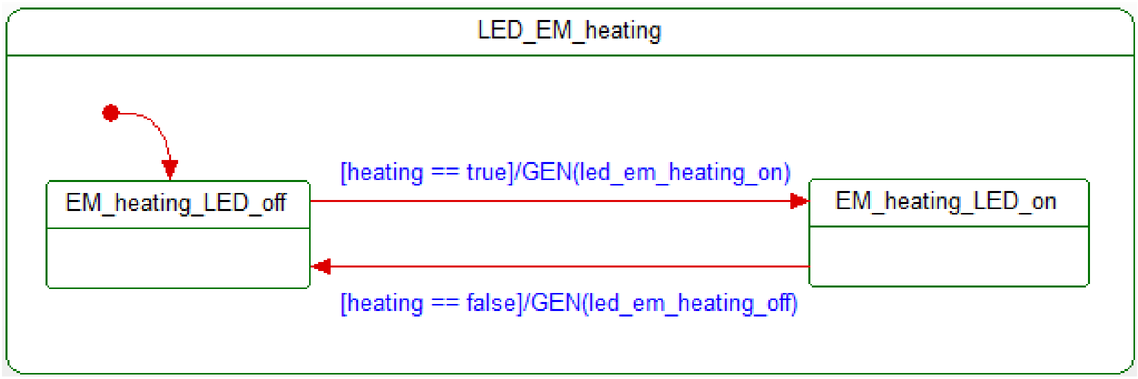

Figure A.16.: 150% Sub State Machine LED_EM_heating

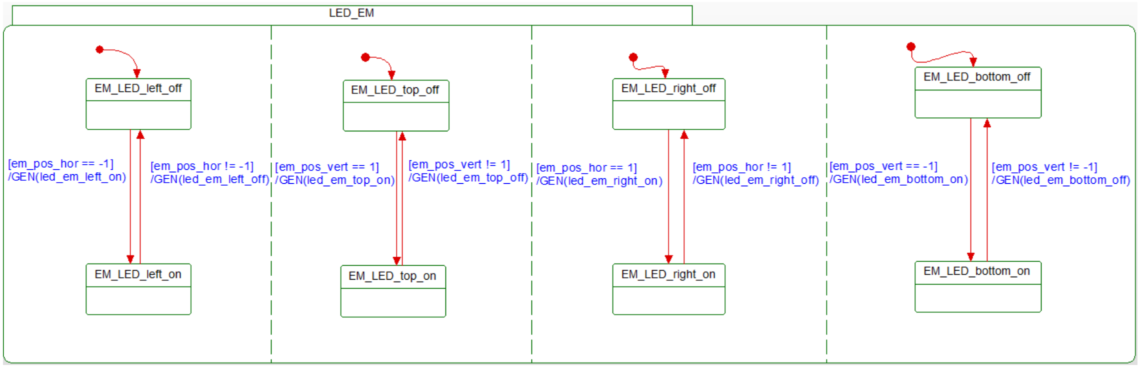

Figure A.17.: 150% Sub State Machine LED_EM

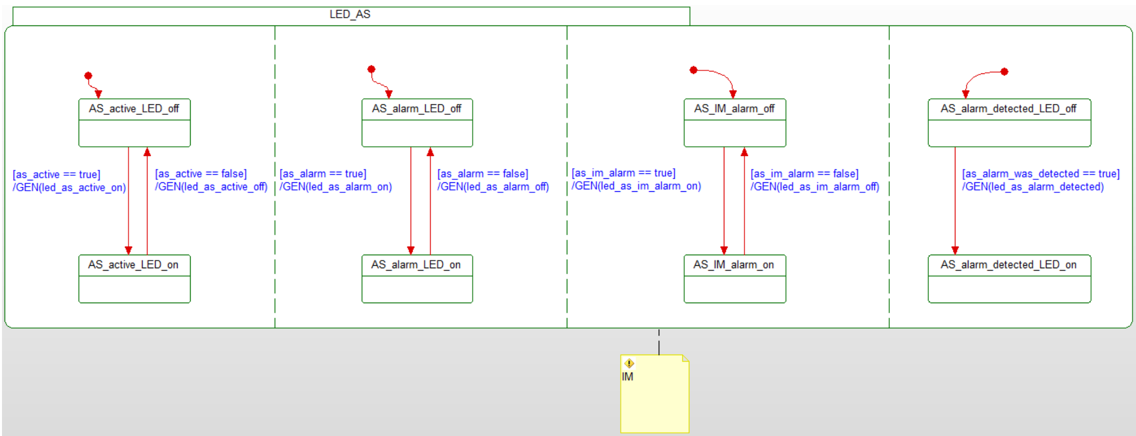

Figure A.18.: 150% Sub State Machine LED_AS

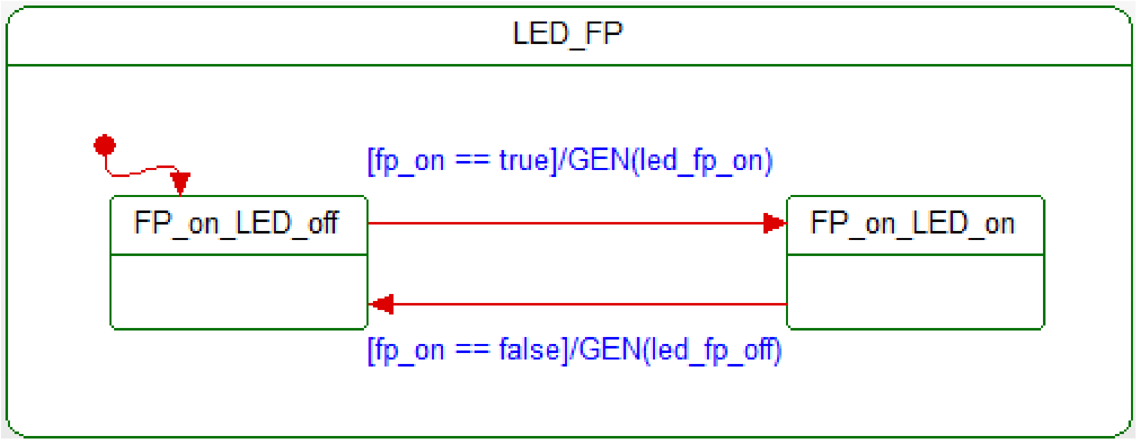

Figure A.19.: 150% Sub State Machine LED_FP

A.2. Definition of the Core Architecture Model in DELTARX

```
architecture BCS for featuremodel 'BCS.featuremodel'{
  signals {
    pw_but_mv_dn boolean
    pw_but_mv_up boolean
    em_but_mv_left boolean
    em_but_mv_right boolean
    em_but_mv_up boolean
    em_but_mv_dn boolean

    pw_but_up boolean
    pw_but_dn boolean
    em_but_right boolean
    em_but_left boolean
    em_but_up boolean
    em_but_down boolean

    em_pos_left boolean
    em_pos_right boolean
    em_pos_top boolean
    em_pos_bottom boolean
    em_mv_left boolean
    em_mv_right boolean
    em_mv_up boolean
    em_mv_down boolean
```

```

    finger_detected boolean
    fp_on boolean
    fp_off boolean

    pw_pos_up boolean
    pw_pos_dn boolean
    pw_mv_up boolean
    pw_mv_dn boolean
  }

  components {
    HMI {
      ports {
        in p_pw_but_mv_dn pw_but_mv_dn
        in p_pw_but_mv_up pw_but_mv_up
        in p_em_but_mv_left em_but_mv_left
        in p_em_but_mv_right em_but_mv_right
        in p_em_but_mv_up em_but_mv_up
        in p_em_but_mv_dn em_but_mv_dn

        out p_pw_but_up pw_but_up
        out p_pw_but_dn pw_but_dn
        out p_em_but_right em_but_right
        out p_em_but_left em_but_left
        out p_em_but_up em_but_up
        out p_em_but_down em_but_down
      }
    }

    ManPW {
      ports {
        in p_pw_but_up pw_but_up
        in p_pw_but_dn pw_but_dn
        in p_pos_up pw_pos_up
        in p_pos_dn pw_pos_dn
        in p_fp_on fp_on
        in p_fp_off fp_off

        out p_pw_mv_dn pw_mv_dn
```

```
      out p_pw_mv_up pw_mv_up
    }
  }

  FP {
    ports {
      in p_finger_detected finger_detected
      in p_pw_but_dn pw_but_dn

      out p_fp_on fp_on
      out p_fp_off fp_off
    }
  }

  EM {
    ports {
      in p_em_but_right em_but_right
      in p_em_but_left em_but_left
      in p_em_but_up em_but_up
      in p_em_but_down em_but_down
      in p_em_pos_left em_pos_left
      in p_em_pos_right em_pos_right
      in p_em_pos_top em_pos_top
      in p_em_pos_bottom em_pos_bottom

      out p_em_mv_left em_mv_left
      out p_em_mv_right em_mv_right
      out p_em_mv_up em_mv_up
      out p_em_mv_dn em_mv_down
    }
  }
}

connectors {
  hmi1(HMI,em_but_right,em_but_right,EM)
  hmi2(HMI,em_but_left,em_but_left,EM)
  hmi3(HMI,em_but_up,em_but_up,EM)
  hmi4(HMI,em_but_down,em_but_down,EM)
  hmi5(HMI,pw_but_up,pw_but_up,ManPW)
  hmi6(HMI,pw_but_dn,pw_but_dn,ManPW)
```

```
    hmi7(HMI,pw_but_dn,pw_but_dn,FP)

    env1(ENV,pw_but_mv_dn,pw_but_mv_dn,HMI)
    env2(ENV,pw_but_mv_up,pw_but_mv_up,HMI)
    env3(ENV,em_but_mv_left,em_but_mv_left,HMI)
    env4(ENV,em_but_mv_right,em_but_mv_right,HMI)
    env5(ENV,em_but_mv_up,em_but_mv_up,HMI)
    env6(ENV,em_but_mv_dn,em_but_mv_dn,HMI)
    env7(ENV,em_pos_left,em_pos_left,EM)
    env8(ENV,em_pos_right,em_pos_right,EM)
    env9(ENV,em_pos_top,em_pos_top,EM)
    env10(ENV,em_pos_bottom,em_pos_bottom,EM)
    env11(ENV,finger_detected,finger_detected,FP)
    env12(ENV,pw_pos_up, pw_pos_up,ManPW)
    env13(ENV,pw_pos_dn,pw_pos_dn,ManPW)

    fp1(FP,fp_on,fp_on,ManPW)
    fp2(FP,fp_off,fp_off,ManPW)

    em1(EM,em_mv_left,em_mv_left,ENV)
    em2(EM,em_mv_right,em_mv_right,ENV)
    em3(EM,em_mv_up,em_mv_up,ENV)
    em4(EM,em_mv_down,em_mv_down,ENV)

    pw1(ManPW,pw_mv_dn,pw_mv_dn,ENV)
    pw2(ManPW,pw_mv_up,pw_mv_up,ENV)
  }
```

Listing A.1: Core Architecture Model P0 in DeltaRx

A.3. BCS System Requirements

A total of 97 requirements have been defined for the Body Comfort System. They describe the required functionality and, thus, represent the system specification. Each requirement has been originally defined in German, i.e., they are written in natural language. These requirements are not defined for different product variants, but for the original single-software system [6]. We present these 97 requirements in the following, grouped after their corresponding system component.

Requirements for Automatic Locking

Requirement AL1

Beim Ueberschreiten einer definierten Geschwindigkeit, werden saemtliche Tueren abgeschlossen, wenn sie nicht abgeschlossen sind.

Requirement AL2

Beim Ueberschreiten einer definierten Geschwindigkeit passiert nichts, wenn die Tueren bereits manuell abgeschlossen wurden.

Requirement AL3

Wird die Geschwindigkeit verringert, sodass sie den definierten Wert unterschreitet und wurden die Tueren durch das Automatic Locking Feature abgeschlossen, werden die Tueren aufgeschlossen.

Requirement AL4

Wird die Geschwindigkeit verringert, sodass sie den definierten Wert unterschreitet und wurden die Tueren nicht durch das Automatic Locking Feature, sondern zuvor manuell abgeschlossen, werden die Tueren nicht aufgeschlossen.

Requirements for Automatic Power Window

Requirement AutPW1

Druecken auf den Knopf zum Oeffnen des Fensters fuehrt dazu, dass das Fenster komplett geoeffnet wird.

Requirement AutPW2

Druecken auf den Knopf zum Schliessen des Fensters fuehrt dazu, dass das Fenster komplett geschlossen wird.

Requirement AutPW3

Wird das Fenster gerade geoeffnet, fuehrt Druecken der Taste zum Schliessen dazu, dass das Fenster anhaelt.

Requirement AutPW4

Wird das Fenster gerade geoeffnet, passiert durch Druecken der Taste zum Oeffnen nichts.

Requirement AutPW5

Wird das Fenster gerade geschlossen, fuehrt Druecken der Taste zum Oeffnen dazu, dass das Fenster anhaelt.

Requirement AutPW6

Wird das Fenster gerade geschlossen, passiert durch Druecken der Taste zum Schliessen nichts.

Requirement AutPW7

Ist das Fenster komplett geoeffnet, hat das Druecken der Taste zum Oeffnen keinen Effekt.

Requirement AutPW8

Ist das Fenster komplett geschlossen, hat das Druecken der Taste zum Schliessen keinen Effekt.

Requirements for Alarm System

Requirement AS1

Gewaltsames Oeffnen einer Tuer fuehrt bei aktiver Alarmanlage zum Ausloesen von Alarm.

Requirement AS2

Gewaltsames Oeffnen einer Tuer fuehrt bei deaktivierter Alarmanlage nicht zum Ausloesen von Alarm.

Requirement AS3

Laufender Alarm wird, falls LEDs vorhanden sind, durch Aufleuchten einer LED signalisiert.

Requirement AS4

Alarm wird automatisch nach 20 Sekunden beendet, woraufhin, falls LEDs vorhanden sind, eine Mitteilung ueber diese angezeigt wird.

Requirement AS5

Alarm kann durch das Aufschliessen des Fahrzeugs deaktiviert werden. Dies fuehrt, wenn LEDs vorhanden sind, nicht zu einer Mitteilung.

Requirements for Control Alarm System

Requirement CAS1

Durch Druck auf den entsprechenden Knopf auf der Fernbedienung, wird die Alarmanlage aktiviert, wenn sie zuvor inaktiv war.

Requirement CAS2

Ist die Alarmanlage aktiv, hat das Druecken der Fernbedienungstaste zum Aktivieren keinen Effekt.

Requirement CAS3

Durch Druck auf den entsprechenden Knopf auf der Fernbedienung, wird die Alarmanlage deaktiviert, wenn sie zuvor aktiv war.

Requirement CAS4

Ist die Alarmanlage nicht aktiv, hat das Druecken der Fernbedienungstaste zum Deaktivieren keinen Effekt.

Requirement for Controlling Automatic Power Window

Requirement CAP1

Druck auf den Fernbedienungsknopf zum Oeffnen des Fensters laesst das Fenster zum Oeffnen nach unten fahren.

Requirement CAP2

Druck auf den Fernbedienungsknopf zum Schliessen des Fensters laesst das Fenster zum Schliessen nach oben fahren.

Requirement CAP3

Wird das Fenster gerade geoeffnet, fuehrt Druecken der Fernbedienungstaste zum Schliessen dazu, dass das Fenster anhaelt.

Requirement CAP4

Wird das Fenster gerade geoeffnet, passiert durch Druecken der Taste zum Oeffnen nichts.

Requirement CAP5

Wird das Fenster gerade geschlossen, fuehrt Druecken der Fernbedienungstaste zum Oeffnen dazu, dass das Fenster anhaelt.

Requirement CAP6

Wird das Fenster gerade geschlossen, passiert durch Druecken der Taste zum Schliessen nichts.

Requirement CAP7

Ist das Fenster komplett geoeffnet, hat das Druecken der Fernbedienungstaste zum Oeffnen keinen Effekt.

Requirement CAP8

Ist das Fenster komplett geschlossen, hat das Druecken der Fernbedienungstaste zum Schliessen keinen Effekt.

Requirement for Central Locking System

Requirement CLS1

Bei Druck auf den Verriegelungsknopf und inaktiver Zentralverriegelung wird die Zentralverriegelung aktiviert, wodurch alle Tueren abgeschlossen werden.

Requirement CLS2

Bei Druck auf den Verriegelungsknopf und aktiver Zentralverriegelung wird keine Aktion ausgeloest.

Requirement CLS3

Bei Druck auf den Entriegelungsknopf und aktiver Zentralverriegelung wird die Zentralverriegelung deaktiviert, wodurch alle Tueren aufgeschlossen werden.

Requirement CLS4

Bei Druck auf den Entriegelungsknopf und inaktiver Zentralverriegelung wird keine Aktion ausgeloest.

Requirement CLS5

Durch Abschliessen einer Tuer bei inaktiver Zentralverriegelung wird die Zentralverriegelung aktiviert, wodurch alle Tueren abgeschlossen werden.

Requirement CLS6

Durch Abschliessen einer Tuer bei aktiver Zentralverriegelung wird keine Aktion ausgeloest.

Requirement CLS7

Durch Aufschliessen einer Tuer bei aktiver Zentralverriegelung wird die Zentralverriegelung deaktiviert, wodurch alle Tueren aufgeschlossen werden.

Requirement CLS8

Durch Aufschliessen einer Tuer bei inaktiver Zentralverriegelung wird keine Aktion ausgeloest.

Requirement CLS9

Bei Verriegelung des Fensters, inaktiver Zentralverriegelung und Existenz des automatischen Fensterhebers werden alle Fenster automatisch geschlossen, falls sie geoeffnet sind. Andernfalls werden sie blockiert.

Requirement CLS10

Bei Entriegelung des Fensters, aktiver Zentralverriegelung und Existenz des automatischen Fensterhebers wird die Blockierung aller Fenster aufgehoben.

Requirement CLS11

Bei Verriegelung des Fensters, inaktiver Zentralverriegelung und Existenz des manuellen Fensterhebers werden alle Fenster blockiert.

Requirement CLS12

Bei Entriegelung des Fensters, aktiver Zentralverriegelung und Existenz des manuellen Fensterhebers wird die Blockierung aller Fenster aufgehoben.

Requirements for Exterior Mirror

Requirement EM1

Wenn der Spiegel eingeklappt ist, bewirkt das Druecken der zentralen Taste, dass der Spiegel ausgeklappt wird.

Requirement EM2

Wenn der Spiegel ausgeklappt ist und das Fahrzeug steht, bewirkt das Druecken der zentralen Taste, dass der Spiegel eingeklappt wird.

Requirement EM3

Wenn der Spiegel ausgeklappt ist und sich das Fahrzeug in Bewegung befindet, hat das Druecken der zentralen Taste keinen Effekt.

Requirement EM4

Wenn die LED vorhanden ist und der Spiegel eingeklappt ist, leuchtet die LED.

Requirement EM5

Wenn die LED vorhanden ist und der Spiegel ausgeklappt ist, leuchtet die LED nicht.

Requirement EM6

Druck auf die Richtungstaste *links* des elektrischen Aussenspiegels bewirkt, dass sich der Aussenspiegel nach links bewegt. Dies funktioniert nur, wenn der Aussenspiegel Bewegungsspielraum nach links hat.

Requirement EM7

Druck auf die Richtungstaste *rechts* des elektrischen Aussenspiegels bewirkt, dass sich der Aussenspiegel nach rechts bewegt. Dies funktioniert nur, wenn der Aussenspiegel Bewegungsspielraum nach rechts hat.

Requirement EM8

Druck auf die Richtungstaste *oben* des elektrischen Aussenspiegels bewirkt, dass sich der Aussenspiegel nach oben bewegt. Dies funktioniert nur, wenn der Aussenspiegel Bewegungsspielraum nach oben hat.

Requirement EM9

Druck auf die Richtungstaste *unten* des elektrischen Aussenspiegels bewirkt, dass sich der Aussenspiegel nach unten bewegt. Dies funktioniert nur, wenn der Aussenspiegel Bewegungsspielraum nach unten hat.

Requirements for Exterior Mirror with Heating

Requirement EMH1

Es sind zwei Temperaturen definiert. Eine minimale Temperatur und eine maximale Temperatur.

Requirement EMH2

Wenn die Termeraturmessung am Aussenspiegel die minimale Temperatur unterschreitet, wird die Aussenspiegelheizung aktiviert.

Requirement EMH3

Abschalten des Fahrzeugs fuehrt zur Abschaltung der Aussenspiegelheizung.

Requirement EMH4

Wenn die Aussenspiegelheizung aktiviert ist und die maximale Temperatur ueberschritten wird, wird die Aussenspiegelheizung deaktiviert.

Requirement EMH5

Wenn die LED vorhanden und die Aussenspiegelheizung aktiviert ist, leuchtet die LED.

Requirement EMH6

Wenn die LED vorhanden und die Aussenspiegelheizung nicht aktiviert ist, leuchtet die LED nicht.

Requirements for Finger Protection

Requirement FP1

Sobald das Fenster geschlossen wird und ein Hindernis im Fensterrahmen durch Gegendruck erkannt wird, wird der Einklemmschutz aktiviert.

Requirement FP2

Ist der Einklemmschutz aktiviert, laesst sich das Fenster nicht mehr schliessen.

Requirement FP3

Der Einklemmschutz laesst sich durch kurzes Druecken der Oeffnen-Taste deaktivieren.

Requirement FP4

Wenn die LED vorhanden und der Einklemmschutz aktiviert ist, leuchtet die LED.

Requirement FP5

Wenn die LED vorhanden und der Einklemmschutz nicht aktiviert ist, leuchtet die LED nicht.

Requirements for Interior Monitoring

Requirement IM1

Bei Aufzeichnung einer Bewegung im Innenraum waehrend die Alarmanlage aktiviert ist, wird Alarm ausgeloest.

Requirement IM2

Die Innenraumueberwachung wird deaktiviert, sobald die Alarmanlage deaktiviert wird.

Requirement IM3

Wenn die LED fuer die Alarmanlage vorhanden ist, leuchtet bei Aktivierung der Innenraumueberwachung eine spezielle LED.

Requirement IM4

Ist die LED fuer die Alarmanlage vorhanden und die Innenraumueberwachung deaktiviert, leuchtet die spezielle LED nicht.

Requirements for LED Alarm System

Requirement LED_AS1

Wenn die Alarmanlage aktiviert ist, wird das durch das Leuchten der LED signalisiert.

Requirement LED_AS2

Wenn die Alarmanlage deaktiviert ist, leuchtet die LED nicht.

Requirement LED_AS3

Wenn ein Alarm ausgeloest wurde, wird das durch das Leuchten der LED signalisiert.

Requirement LED_AS4

Wenn kein Alarm ausgeloest wurde, leuchtet die LED nicht.

Requirement LED_AS5

Wenn die Innenraumueberwachung aktiviert ist, wird das durch das Leuchten der LED signalisiert.

Requirement LED_AS6

Wenn die Innenraumueberwachung nicht aktiviert ist, leuchtet die LED nicht.

Requirement LED_AS7

Die LED, die ausgeloesten Alarm signalisiert, kann nur durch das Druecken der Reset-Taste, die sich neben der LED befindet, zurueckgesetzt werden. Die Taste hat nur einen Effekt, wenn der Zuendschluessel in der Zuendung steckt.

Requirements for LED Central Locking System

Requirement LED_CLS1

Wenn die Zentralverriegelung aktiv und das Fahrzeug somit abgeschlossen ist, leuchtet die LED.

Requirement LED_CLS2

Wenn die Zentralverriegelung inaktiv und das Fahrzeug somit nicht abgeschlossen ist, leuchtet die LED nicht.

Requirement LED_CLS3

Durch das Deaktivieren der Zentralverriegelung wird ebenfalls die LED abgeschalten.

Requirements for LED Exterior Mirror

Requirement LED_EM1

Die LED leuchtet, wenn der Aussenspiegel eingeklappt ist.

Requirement LED_EM2

Die LED leuchtet nicht, wenn der Aussenspiegel ausgeklappt ist.

Requirement LED_EM3

Wechsel zwischen den Zustaenden des Aussenspiegels fuehrt zum Wechsel zwischen den Zustaenden der LED.

Requirement LED_EM4

Eine spezielle LED fuer jede Richtung leuchtet, wenn sich der Aussenspiegel in der maximalen Position in dieser Richtung befindet.

Requirements for LED Exterior Mirror with Heating

Requirement LED_EMH1

Wenn die Aussenspiegelheizung aktiviert ist, leuchtet die LED.

Requirement LED_EMH2

Wenn die Aussenspiegelheizung deaktiviert ist, leuchtet die LED nicht.

Requirement LED_EMH3

Deaktivieren der Aussenspiegelheizung fuehrt zur Deaktivierung der LED.

Requirements for LED Power Window

Requirement LED_PW1

Wenn alle Fenster still stehen leuchtet die LED nicht.

Requirement LED_PW2

Wenn der Knopf zum Schliessen eines Fensters gedrueckt wird und das entsprechende Fenster nach oben faehrt, zeigt die LED an, dass sich ein Fenster schliesst.

Requirement LED_PW3

Wenn der Knopf zum Oeffnen eines Fensters gedrueckt wird und das entsprechende Fenster nach unten fahren, zeigt die LED an, dass sich ein Fenster oeffnet.

Requirement LED_PW4

Beendigung des Oeffnungs- bzw. Schliessvorgangs der Fenster fuehrt zur Deaktivierung der LED.

Requirements for Manual Power Window

Requirement ManPW1

Druecken auf den Knopf zum Schliessen des Fensters fuehrt dazu, dass das Fenster solange geschlossen wird, wie der Druck ausgeuebt wird.

Requirement ManPW2

Druecken auf den Knopf zum Oeffnen des Fensters fuehrt dazu, dass das Fenster solange geoeffnet wird, wie der Druck ausgeuebt wird.

Requirements for Remote Control Key

Requirement RCK1

Druecken auf den Knopf zum Schliessen des Fensters fuehrt dazu, dass das Fenster solange geschlossen wird, wie der Druck ausgeuebt wird.

Requirement RCK2

Druck auf den Fernbedienungsknopf zum Entriegeln deaktiviert die Zentralverriegelung, wenn die Zentralverriegelung aktiviert ist.

Requirement RCK3

Druck auf den Fernbedienungsknopf zum Verriegeln hat keinen Effekt, wenn die Zentralverriegelung aktiv ist.

Requirement RCK4

Druck auf den Fernbedienungsknopf zum Entriegeln hat keinen Effekt, wenn die Zentralverriegelung dekativiert ist.

Requirements for Safety Function

Requirement SF1

Das Verstreichen von zehn Sekunden zwischen dem Aufschliessen und dem Oeffnen einer Tuer fuehrt dazu, dass die Tuer erneut abgeschlossen wird.

Requirement SF2

Das Aufschliessen und Oeffnen einer Tuer innerhalb von 10 Sekunden bewirkt, dass die Tuer nicht wieder automatisch verschlossen wird.

A.4. BCS System Test Cases

A total of 128 test cases have been defined for the BCS system. They are not variant specific, but cover certain functionalities of the system, such as the manual power window or remote control key. Each test case is defined in three parts. First, a precondition is defined, which is a state the system has to be in to successfully execute the test case. Second, an action is defined, which is (manually) executed to perform the test case. Third, each test case has an expected result which is based on the system requirements. If the observable results are not identical to the expected result, we consider the test case to be failed, i.e., the tested functionality has not been implemented correctly.

In the following, we present all 128 test cases for the BCS system, grouped by their tested functionality. The test cases are defined in German language, which is the original language of the BCS specification as well.

System Test Cases for Feature Automatic Locking

Table A.1.: System Test Case AL 1

Step	*Content*
Precondition	Das Fahrzeug faehrt mit einer Geschwindigkeit, die niedriger ist, als die definierte Geschwindigkeit und die Tueren sind nicht abgeschlossen.
Action	Die Geschwindigkeit wird in dem Masse erhoeht, dass die definierte Geschwindigkeit ueberschritten wird.
Expected Result	Die Tueren werden automatisch abgeschlossen.

Table A.2.: System Test Case AL 2

Step	*Content*
Precondition	Das Fahrzeug faehrt mit einer Geschwindigkeit, die niedriger als die definierte Geschwindigkeit ist und die Tueren sind abgeschlossen.
Action	Die Geschwindigkeit in dem Masse erhoeht, dass die definierte Geschwindigkeit ueberschritten wird.
Expected Result	Die Tueren bleiben abgeschlossen.

Table A.3.: System Test Case AL 3

Step	*Content*
Precondition	Das Fahrzeug faehrt mit einer Geschwindigkeit, die hoeher ist, als die definierte Geschwindigkeit und die Tueren sind durch das Automatic Locking Feature abgeschlossen worden.
Action	Die Geschwindigkeit in dem Masse verringert, dass die definierte Geschwindigkeit unterschritten wird.
Expected Result	Die Tueren werden automatisch aufgeschlossen.

Table A.4.: System Test Case AL 4

Step	*Content*
Precondition	Das Fahrzeug faehrt mit einer Geschwindigkeit, die hoeher ist, als die definierte Geschwindigkeit und die Tueren sind manuell abgeschlossen worden.
Action	Die Geschwindigkeit wird in dem Masse verringert, dass die definierte Geschwindigkeit unterschritten wird.
Expected Result	Die Tueren bleiben abgeschlossen.

System Test Cases for Alarm System

Table A.5.: System Test Case AS 1

Step	*Content*
Precondition	Tueren sind verschlossen und die Alarmanlage ist aktiviert.
Action	Eine Tuer wird gewaltsam geoeffnet.
Expected Result	Alarm wird ausgeloest.

Table A.6.: System Test Case AS 2

Step	*Content*
Precondition	Tueren sind verschlossen und die Alarmanlage ist nicht aktiviert.
Action	Eine Tuer wird gewaltsam geoeffnet.
Expected Result	Es passiert nichts.

Table A.7.: System Test Case AS 3

Step	*Content*
Precondition	LEDs sind installiert und die Alarmanlage ist aktiviert.
Action	Alarm wird ausgeloest.
Expected Result	Alarm wird durch leuchtendes LED angezeigt.

Table A.8.: System Test Case AS 4

Step	*Content*
Precondition	LEDs sind in das System eingebunden.
Action	Alarm wird ausgeloest und es vergehen 20 Sekunden.
Expected Result	Der Alarm wird automatisch abgeschaltet und durch das Aufleuchten einer LED signalisiert.

Table A.9.: System Test Case AS 5

Step	*Content*
Precondition	LEDs sind nicht in das System eingebunden.
Action	Alarm wird ausgeloest und es vergehen 20 Sekunden.
Expected Result	Der Alarm wird abgeschaltet.

Table A.10.: System Test Case AS 6

Step	*Content*
Precondition	Alarm laeuft und LEDs sind in das System eingebunden.
Action	Das Fahrzeug wird aufgeschlossen.
Expected Result	Der Alarm wird abgeschaltet und wird nicht durch das Aufleuchten einer LED signalisiert.

Table A.11.: System Test Case AS 7

Step	*Content*
Precondition	Alarm laeuft und LEDs sind nicht in das System eingebunden.
Action	Das Fahrzeug wird aufgeschlossen.
Expected Result	Der Alarm wird abgeschaltet.

System Test Cases for Automatic Power Window

Table A.12.: System Test Case AutPW 1

Step	*Content*
Precondition	Die Fenster sind nicht komplett geoeffnet.
Action	Die Taste zum Oeffnen eines Fensters wird einmalig angetippt.
Expected Result	Das Fenster oeffnet sich komplett.

Table A.13.: System Test Case AutPW 2

Step	*Content*
Precondition	Die Fenster sind nicht komplett geschlossen.
Action	Die Taste zum Schliessen eines Fensters wird einmalig angetippt.
Expected Result	Das Fenster schliesst sich komplett.

Table A.14.: System Test Case AutPW 3

Step	*Content*
Precondition	Ein Fenster wird gerade geoeffnet.
Action	Die Taste zum Schliessen des Fensters wird angetippt.
Expected Result	Das Fenster haelt an.

Table A.15.: System Test Case AutPW 4

Step	*Content*
Precondition	Ein Fenster wird gerade geschlossen.
Action	Die Taste zum Oeffnen des Fensters wird angetippt.
Expected Result	Das Fenster haelt an.

Table A.16.: System Test Case AutPW 5

Step	*Content*
Precondition	Ein Fenster wird gerade geoeffnet.
Action	Die Taste zum Oeffnen des Fensters wird angetippt.
Expected Result	Das Fenster oeffnet sich weiterhin.

Table A.17.: System Test Case AutPW 6

Step	*Content*
Precondition	Ein Fenster wird gerade geschlossen.
Action	Die Taste zum Schliessen des Fensters wird angetippt.
Expected Result	Das Fenster schliesst sich weiterhin.

Table A.18.: System Test Case AutPW 7

Step	*Content*
Precondition	Ein Fenster ist komplett geoeffnet.
Action	Die Taste zum Oeffnen wird angetippt.
Expected Result	Das Fenster bleibt komplett geoeffnet.

Table A.19.: System Test Case AutPW 8

Step	*Content*
Precondition	Ein Fenster ist komplett geschlossen.
Action	Die Taste zum Schliessen wird angetippt.
Expected Result	Das Fenster bleibt komplett geschlossen.

System Test Cases for Control Automatic Power Window

Table A.20.: System Test Case CAP 1

Step	*Content*
Precondition	Die Fenster sind nicht komplett geoeffnet.
Action	Die Fernbedienungstaste zum Oeffnen des Fensters wird angetippt.
Expected Result	Das Fenster oeffnet sich komplett.

Table A.21.: System Test Case CAP 2

Step	*Content*
Precondition	Die Fenster sind nicht komplett geschlossen.
Action	Die Fernbedienungstaste zum Schliessen des Fensters wird angetippt.
Expected Result	Das Fenster schliesst sich komplett.

Table A.22.: System Test Case CAP 3

Step	*Content*
Precondition	Ein Fenster oeffnet sich gerade.
Action	Die Fernbedienungstaste zum Schliessen des Fensters wird angetippt.
Expected Result	Das Fenster haelt an.

Table A.23.: System Test Case CAP 4

Step	*Content*
Precondition	Ein Fenster schliesst sich gerade.
Action	Die Fernbedienungstaste zum Oeffnen des Fensters wird angetippt.
Expected Result	Das Fenster haelt an.

Table A.24.: System Test Case CAP 5

Step	*Content*
Precondition	Ein Fenster oeffnet sich gerade.
Action	Die Fernbedienungstaste zum Oeffnen des Fensters wird angetippt.
Expected Result	Das Fenster oeffnet sich weiterhin.

Table A.25.: System Test Case CAP 6

Step	*Content*
Precondition	Ein Fenster schliesst sich gerade.
Action	Die Fernbedienungstaste zum Schliessen des Fensters wird angetippt.
Expected Result	Das Fenster schliesst sich weiterhin.

Table A.26.: System Test Case CAP 7

Step	*Content*
Precondition	Die Fenster sind komplett geoeffnet.
Action	Die Fernbedienungstaste zum Oeffnen des Fensters wird angetippt.
Expected Result	Das Fenster bewegt sich nicht.

Table A.27.: System Test Case CAP 8

Step	*Content*
Precondition	Die Fenster sind komplett geschlossen.
Action	Die Fernbedienungstaste zum Schliessen des Fensters wird angetippt.
Expected Result	Das Fenster bewegt sich nicht.

System Test Cases for Control Alarm System

Table A.28.: System Test Case CAS 1

Step	*Content*
Precondition	Die Alarmanlage ist deaktiviert.
Action	Der Fernbedienungsknopf zum Aktivieren der Alarmanlage wird gedrueckt.
Expected Result	Die Alarmanlage wird aktiviert.

Table A.29.: System Test Case CAS 2

Step	*Content*
Precondition	Die Alarmanlage ist aktiviert.
Action	Der Fernbedienungsknopf zum Aktivieren der Alarmanlage wird gedrueckt.
Expected Result	Die Alarmanlage bleibt aktiviert.

Table A.30.: System Test Case CAS 3

Step	*Content*
Precondition	Die Alarmanlage ist aktiviert.
Action	Der Fernbedienungsknopf zum Deaktivieren der Alarmanlage wird gedrueckt.
Expected Result	Die Alarmanlage wird deaktiviert.

Table A.31.: System Test Case CAS 4

Step	*Content*
Precondition	Die Alarmanlage ist deaktiviert.
Action	Der Fernbedienungsknopf zum Deaktivieren der Alarmanlage wird gedrueckt.
Expected Result	Die Alarmanlage bleibt deaktiviert.

System Test Cases for Central Locking System

Table A.32.: System Test Case CLS 1

Step	*Content*
Precondition	Die Zentralverriegelung ist inaktiv.
Action	Der Verriegelungsknopf wird gedrueckt.
Expected Result	Die Zentralverriegelung wird aktiviert und alle Tueren werden verschlossen.

Table A.33.: System Test Case CLS 2

Step	*Content*
Precondition	Die Zentralverriegelung ist aktiv.
Action	Der Verriegelungsknopf wird gedrueckt.
Expected Result	Die Zentralverriegelung bleibt aktiv.

Table A.34.: System Test Case CLS 3

Step	*Content*
Precondition	Die Zentralverriegelung ist aktiv.
Action	Der Entriegelungsknopf wird gedrueckt.
Expected Result	Die Zentralverriegelung wird deaktiviert und alle Tueren werden aufgeschlossen.

Table A.35.: System Test Case CLS 4

Step	*Content*
Precondition	Die Zentralverriegelung ist inaktiv.
Action	Der Entriegelungsknopf wird gedrueckt.
Expected Result	Die Zentralverriegelung bleibt inaktiv.

Table A.36.: System Test Case CLS 5

Step	*Content*
Precondition	Die Zentralverriegelung ist inaktiv.
Action	Eine Tuer wird abgeschlossen.
Expected Result	Die Zentralverriegelung wird aktiviert und alle Tueren werden verschlossen.

Table A.37.: System Test Case CLS 6

Step	*Content*
Precondition	Die Zentralverriegelung ist aktiv.
Action	Eine Tuer wird abgeschlossen.
Expected Result	Die Zentralverriegelung bleibt aktiv.

Table A.38.: System Test Case CLS 7

Step	*Content*
Precondition	Die Zentralverriegelung ist aktiviert.
Action	Eine Tuer wird aufgeschlossen.
Expected Result	Die Zentralverriegelung wird deaktiviert und alle Tueren werden aufgeschlossen.

Table A.39.: System Test Case CLS 8

Step	*Content*
Precondition	Die Zentralverriegelung ist inaktiv.
Action	Eine Tuer wird aufgeschlossen.
Expected Result	Die Zentralverriegelung bleibt inaktiv.

Table A.40.: System Test Case CLS 9

Step	*Content*
Precondition	Die Zentralverriegelung ist inaktiv, automatische Fensterheber sind im System vorhanden und die Fenster sind geoeffnet.
Action	Die Zentralverriegelung wird aktiviert.
Expected Result	Die Fenster schliessen sich komplett und werden im Anschluss verriegelt.

Table A.41.: System Test Case CLS 10

Step	*Content*
Precondition	Die Zentralverriegelung ist inaktiv, automatische Fensterheber sind im System vorhanden und die Fenster sind geschlossen.
Action	Die Zentralverriegelung wird aktiviert.
Expected Result	Die Fenster werden blockiert.

Table A.42.: System Test Case CLS 11

Step	*Content*
Precondition	Die Zentralverriegelung ist aktiv und automatische Fensterheber sind im System vorhanden.
Action	Die Zentralverriegelung wird deaktiviert.
Expected Result	Die Fenster werden nicht mehr blockiert.

Table A.43.: System Test Case CLS 12

Step	*Content*
Precondition	Die Zentralverriegelung ist inaktiv, manuelle Fensterheber sind im System vorhanden und die Fenster sind geoeffnet.
Action	Die Zentralverriegelung wird aktiviert.
Expected Result	Die Fenster werden blockiert.

Table A.44.: System Test Case CLS 13

Step	*Content*
Precondition	Die Zentralverriegelung ist inaktiv, manuelle Fensterheber sind im System vorhanden und die Fenster sind geschlossen.
Action	Die Zentralverriegelung wird aktiviert.
Expected Result	Die Fenster werden blockiert.

Table A.45.: System Test Case CLS 14

Step	*Content*
Precondition	Die Zentralverriegelung ist aktiv und manuelle Fensterheber sind im System vorhanden.
Action	Die Zentralverriegelung wird deaktiviert.
Expected Result	Die Fenster werden nicht mehr blockiert.

System Test Cases for Exterior Mirror

Table A.46.: System Test Case EM 1

Step	*Content*
Precondition	Der Aussenspiegel ist eingeklappt.
Action	Die zentrale Taste der Aussenspiegelsteuerung wird betaetigt.
Expected Result	Der Aussenspiegel wird ausgeklappt

Table A.47.: System Test Case EM 2

Step	*Content*
Precondition	Der Aussenspiegel ist ausgeklappt und das Fahrzeug steht.
Action	Die zentrale Taste der Aussenspiegelsteuerung wird betaetigt.
Expected Result	Der Aussenspiegel wird eingeklappt.

Table A.48.: System Test Case EM 3

Step	*Content*
Precondition	Der Aussenspiegel ist ausgeklappt und das Fahrzeug ist in Bewegung.
Action	Die zentrale Taste der Aussenspiegelsteuerung wird betaetigt.
Expected Result	Der Aussenspiegel wird nicht eingeklappt.

Table A.49.: System Test Case EM 4

Step	*Content*
Precondition	Der Aussenspiegel ist eingeklappt und die LED ist vorhanden.
Action	Die elektrischen Geraete werden durch Drehen des Zuendschluessels aktiviert.
Expected Result	Die LED zeigt durch Leuchten an, dass der Aussenspiegel eingeklappt ist.

Table A.50.: System Test Case EM 5

Step	*Content*
Precondition	Der Aussenspiegel ist eingeklappt und die LED ist vorhanden. Die LED signalisiert, dass der Aussenspiegel eingeklappt ist.
Action	Die zentrale Taste der Aussenspiegelsteuerung wird betaetigt.
Expected Result	Die LED, die den eingeklappten Zustand des Aussenspiegels signalisiert, erlischt.

Table A.51.: System Test Case EM 6

Step	*Content*
Precondition	Die Elektronik des Fahrzeugs ist aktiviert und der Aussenspiegel hat potentiellen Bewegungsspielraum nach links.
Action	Die Richtungstaste *links* wird betaetigt.
Expected Result	Der Aussenspiegel bewegt sich nach links.

Table A.52.: System Test Case EM 7

Step	*Content*
Precondition	Die Elektronik des Fahrzeugs ist aktiviert und der Aussenspiegel hat keinen potentiellen Bewegungsspielraum nach links
Action	Die Richtungstaste *links* wird betaetigt.
Expected Result	Der Aussenspiegel bewegt sich nicht.

Table A.53.: System Test Case EM 8

Step	*Content*
Precondition	Die Elektronik des Fahrzeugs ist aktiviert und der Aussenspiegel hat potentiellen Bewegungsspielraum nach rechts.
Action	Die Richtungstaste *rechts* wird betaetigt.
Expected Result	Der Aussenspiegel bewegt sich nach rechts.

Table A.54.: System Test Case EM 9

Step	*Content*
Precondition	Die Elektronik des Fahrzeugs ist aktiviert und der Aussenspiegel hat keinen keinen potentiellen Bewegungsspielraum nach links.
Action	Die Richtungstaste *rechts* wird betaetigt.
Expected Result	Der Aussenspiegel bewegt sich nicht.

Table A.55.: System Test Case EM 10

Step	*Content*
Precondition	Die Elektronik des Fahrzeugs ist aktiviert und der Aussenspiegel hat potentiellen Bewegungsspielraum nach oben.
Action	Die Richtungstaste *oben* wird betaetigt.
Expected Result	Der Aussenspiegel bewegt sich nach oben.

Table A.56.: System Test Case EM 11

Step	*Content*
Precondition	Die Elektronik des Fahrzeugs ist aktiviert und der Aussenspiegel hat keinen potentiellen Bewegungsspielraum nach oben.
Action	Die Richtungstaste *oben* wird betaetigt.
Expected Result	Der Aussenspiegel bewegt sich nicht.

Table A.57.: System Test Case EM 12

Step	*Content*
Precondition	Die Elektronik des Fahrzeugs ist aktiviert und der Aussenspiegel hat potentiellen Bewegungsspielraum nach unten.
Action	Die Richtungstaste *unten* wird betaetigt.
Expected Result	Der Aussenspiegel bewegt sich nach unten.

Table A.58.: System Test Case EM 13

Step	*Content*
Precondition	Die Elektronik des Fahrzeugs ist aktiviert und der Aussenspiegel hat keinen potentiellen Bewegungsspielraum nach unten.
Action	Die Richtungstaste *unten* wird betaetigt.
Expected Result	Der Aussenspiegel bewegt sich nicht.

System Test Cases for Finger Protection

Table A.59.: System Test Case FP 1

Step	*Content*
Precondition	Ein Hindernis befindet sich im Fenster.
Action	Das Fenster wird geschlossen.
Expected Result	Das Hindernis wird erkannt und der Einklemmschutz wird aktiviert.

Table A.60.: System Test Case FP 2

Step	*Content*
Precondition	Der Einklemmschutz ist aktiviert.
Action	Die Taste zum Schliessen des Fensters wird betaetigt.
Expected Result	Das Fenster schliesst sich nicht.

Table A.61.: System Test Case FP 3

Step	*Content*
Precondition	Der Einklemmschutz ist aktiviert.
Action	Die Taste zum Oeffnen des Fensters wird betaetigt.
Expected Result	Der Einklemmschutz wird deaktiviert.

Table A.62.: System Test Case FP 4

Step	*Content*
Precondition	Die LED fuer den Einklemmschutz ist im Fahrzeug vorhanden.
Action	Der Einklemmschutz wird aktiviert.
Expected Result	Die LED signalisiert den aktivierten Einklemmschutz durch Leuchten.

Table A.63.: System Test Case FP 5

Step	*Content*
Precondition	Die LED fuer den Einklemmschutz ist im Fahrzeug vorhanden.
Action	Der Einklemmschutz wird nicht aktiviert.
Expected Result	Die LED leuchtet nicht.

Table A.64.: System Test Case FP 6

Step	*Content*
Precondition	Die LED fuer den Einklemmschutz ist im Fahrzeug vorhanden und der Einklemmschutz ist aktiviert. Die LED leuchtet.
Action	Der Einklemmschutz wird deaktiviert.
Expected Result	Die LED hoert auf zu leuchten.

System Test Cases for Exterior Mirror with Heating

Table A.65.: System Test Case EMH 1

Step	*Content*
Precondition	Die Aussenspiegelheizung ist im System integriert.
Action	Auslesen der Systemkonfiguration durch externes Geraet.
Expected Result	Bestaetigung der Definition von minimaler und maximaler Temperatur.

Table A.66.: System Test Case EMH 2

Step	*Content*
Precondition	Die Aussenspiegelheizung ist nicht aktiviert.
Action	Am Aussenspiegel wird eine Temperatur unter dem minimalen Wert gemessen.
Expected Result	Die Aussenspiegelheizung wird aktiviert.

Table A.67.: System Test Case EMH 3

Step	*Content*
Precondition	Die Aussenspiegelheizung ist aktiviert.
Action	Das Fahrzeug wird abgeschaltet.
Expected Result	Die Aussenspiegelheizung wird deaktiviert.

Table A.68.: System Test Case EMH 4

Step	*Content*
Precondition	Die Aussenspiegelheizung ist aktiviert.
Action	Am Aussenspiegel wird ein Wert ueber der maximalen Temperatur gemessen.
Expected Result	Die Aussenspiegelheizung wird deaktiviert.

Table A.69.: System Test Case EMH 5

Step	*Content*
Precondition	Die LED fuer die Aussenspiegelheizung ist vorhanden.
Action	Die Aussenspiegelheizung wird aktiviert.
Expected Result	Die LED leuchtet

Table A.70.: System Test Case EMH 6

Step	*Content*
Precondition	Die LED fuer die Aussenspiegelheizung ist vorhanden.
Action	Die Aussenspiegelheizung wird nicht aktiviert.
Expected Result	Die LED leuchtet nicht.

Table A.71.: System Test Case EMH 7

Step	*Content*
Precondition	Die LED fuer die Aussenspiegelheizung ist vorhanden und die Aussenspiegelheizung ist aktiviert. Die LED leuchtet.
Action	Die Aussenspiegelheizung wird deaktiviert.
Expected Result	Die LED hoert auf zu leuchten.

System Test Cases for Interior Monitor

Table A.72.: System Test Case IM 1

Step	*Content*
Precondition	Die Alarmanlage ist aktiviert.
Action	Eine Bewegung im Innenraum wird registriert.
Expected Result	Alarm wird ausgeloest.

Table A.73.: System Test Case IM 2

Step	*Content*
Precondition	Die Alarmanlage und die Innenraumueberwachung sind aktiviert.
Action	Die Alarmanlage wird deaktiviert.
Expected Result	Die Innenraumueberwachung wird deaktiviert.

Table A.74.: System Test Case IM 3

Step	*Content*
Precondition	Die LED ist vorhanden.
Action	Die Innenraumueberwachung wird aktiviert.
Expected Result	Das Aktivieren der Innenraumueberwachung wird mit einer gesonderten LED signalisiert.

Table A.75.: System Test Case IM 4

Step	*Content*
Precondition	Die LED ist vorhanden und die Innenraumueberwachung ist nicht aktiviert. Die LED leuchtet nicht.
Action	Die Innenraumueberwachung wird nicht aktiviert.
Expected Result	Die LED leuchtet nicht.

Table A.76.: System Test Case IM 5

Step	*Content*
Precondition	Die LED ist vorhanden und die Innenraumueberwachung ist aktiviert. Die LED leuchtet.
Action	Die Innenraumueberwachung wird deaktiviert.
Expected Result	Die LED hoert auf zu leuchten.

System Test Cases for LED Alarm System

Table A.77.: System Test Case LED_AS 1

Step	*Content*
Precondition	Die LED ist vorhanden.
Action	Die Alarmanlage wird aktiviert.
Expected Result	Die LED leuchtet.

Table A.78.: System Test Case LED_AS 2

Step	*Content*
Precondition	Die LED ist vorhanden und die Alarmanlage ist nicht aktiviert. Die LED leuchtet nicht.
Action	Die Alarmanlage wird nicht aktiviert.
Expected Result	Die LED leuchtet nicht.

Table A.79.: System Test Case LED_AS 3

Step	*Content*
Precondition	Die LED ist vorhanden und die Alarmanlage ist aktiviert. Die LED leuchtet.
Action	Die Alarmanlage wird deaktiviert.
Expected Result	Die LED leuchtet nicht.

Table A.80.: System Test Case LED_AS 4

Step	*Content*
Precondition	Die Alarmanlage ist aktiviert und die LED ist vorhanden.
Action	Es wird Alarm ausgeloest.
Expected Result	Die LED zum signalisieren des Alarms leuchtet.

Table A.81.: System Test Case LED_AS 5

Step	*Content*
Precondition	Die Alarmanlage ist aktiviert und die LED ist vorhanden.
Action	Es wird kein Alarm ausgeloest.
Expected Result	Die LED zum signalisieren des Alarms leuchtet nicht.

Table A.82.: System Test Case LED_AS 6

Step	*Content*
Precondition	Die LED ist vorhanden und die Alarmanlage ist aktiviert.
Action	Die Innenraumueberwachung wird aktiviert.
Expected Result	Die LED zum Signalisieren der Innenraumueberwachung leuchtet.

Table A.83.: System Test Case LED_AS 7

Step	*Content*
Precondition	Die LED ist vorhanden und die Alarmanlage ist aktiviert.
Action	Die Innenraumueberwachung wird nicht aktiviert.
Expected Result	Die LED zum Signalisieren der Innenraumueberwachung leuchtet nicht.

Table A.84.: System Test Case LED_AS 8

Step	*Content*
Precondition	Die LED ist vorhanden, die Alarmanlage und die Innenraumueberwachung sind aktiviert.
Action	Die Innenraumueberwachung wird deaktiviert.
Expected Result	Die LED zum Signalisieren der Innenraumueberwachung leuchtet nicht.

Table A.85.: System Test Case LED_AS 9

Step	*Content*
Precondition	Ein Alarm wurde ausgeloest und die LED zum Signalisieren des Alarms ist aktiv.
Action	Der Reset-Knopf wird gedrueckt, waehrend sich der Schlu-essel im Schloss befindet.
Expected Result	Die LED erlischt.

Table A.86.: System Test Case LED_AS 10

Step	*Content*
Precondition	Ein Alarm wurde ausgeloest und die LED zum Signalisieren des Alarms ist aktiv.
Action	Der Reset-Knopf wird gedrueckt, waehrend sich der Schlu-essel nicht im Schloss befindet.
Expected Result	Die LED leuchtet weiterhin.

Test Cases for LED Central Locking System

Table A.87.: System Test Case LED_CLS 1

Step	*Content*
Precondition	Die LED ist vorhanden.
Action	Die Zentralverriegelung wird aktiviert.
Expected Result	Die LED leuchtet.

Table A.88.: System Test Case LED_CLS 2

Step	*Content*
Precondition	Die LED ist vorhanden.
Action	Die Zentralverriegelung wird nicht aktiviert.
Expected Result	Die LED leuchtet nicht.

Table A.89.: System Test Case LED_CLS 3

Step	*Content*
Precondition	Die LED ist vorhanden und die Zentralverriegelung ist aktiviert. Die LED leuchtet.
Action	Die Zentralverriegelung wird deaktiviert.
Expected Result	Die LED hoert auf zu leuchten.

Test Cases for LED Exterior Mirror

Table A.90.: System Test Case LED_EM 1

Step	*Content*
Precondition	Der Aussenspiegel ist eingeklappt.
Action	Das Fahrzeug wird gestartet.
Expected Result	Die LED leuchtet.

Table A.91.: System Test Case LED_EM 2

Step	*Content*
Precondition	Der Aussenspiegel ist ausgeklappt.
Action	Das Fahrzeug wird gestartet.
Expected Result	Die LED leuchtet nicht.

Table A.92.: System Test Case LED_EM 3

Step	*Content*
Precondition	Der Spiegel ist eingeklappt. Die LED leuchtet.
Action	Der Spiegel wird ausgeklappt.
Expected Result	Die LED hoert auf zu leuchten.

Table A.93.: System Test Case LED_EM 4

Step	*Content*
Precondition	Der Spiegel ist ausgeklappt. Die LED leuchtet nicht.
Action	Der Spiegel wird eingeklappt.
Expected Result	Die LED beginnt zu leuchten.

Table A.94.: System Test Case LED_EM 5

Step	*Content*
Precondition	Der Spiegel ist frei in alle Richtungen bewegbar.
Action	Der Spiegel wird nach links bewegt und erreicht seine maximal linke Position.
Expected Result	Eine LED, die das Ende der Bewegungsfreiheit in diese Richtung anzeigt, beginnt zu leuchten.

Table A.95.: System Test Case LED_EM 6

Step	*Content*
Precondition	Der Spiegel ist frei in alle Richtungen bewegbar.
Action	Der Spiegel wird nach rechts bewegt und erreicht seine maximal rechte Position.
Expected Result	Eine LED, die das Ende der Bewegungsfreiheit in diese Richtung anzeigt, beginnt zu leuchten.

Table A.96.: System Test Case LED_EM 7

Step	*Content*
Precondition	Der Spiegel ist frei in alle Richtungen bewegbar.
Action	Der Spiegel wird nach oben bewegt und erreicht seine maximal obere Position.
Expected Result	Eine LED, die das Ende der Bewegungsfreiheit in diese Richtung anzeigt, beginnt zu leuchten.

Table A.97.: System Test Case LED_EM 8

Step	*Content*
Precondition	Der Spiegel ist frei in alle Richtungen bewegbar.
Action	Der Spiegel wird nach unten bewegt und erreicht seine maximal untere Position.
Expected Result	Eine LED, die das Ende der Bewegungsfreiheit in diese Richtung anzeigt, beginnt zu leuchten.

Table A.98.: System Test Case LED_EM 9

Step	*Content*
Precondition	Der Spiegel befindet sich in seiner maximal linken Position. Dies wird durch das Leuchten der LED signalisiert.
Action	Der Spiegel wird nach rechts bewegt.
Expected Result	Die LED hoert auf zu leuchten.

Table A.99.: System Test Case LED_EM 10

Step	*Content*
Precondition	Der Spiegel befindet sich in seiner maximal rechten Position. Dies wird durch das Leuchten der LED signalisiert.
Action	Der Spiegel wird nach links bewegt.
Expected Result	Die LED hoert auf zu leuchten.

Table A.100.: System Test Case LED_EM 11

Step	*Content*
Precondition	Der Spiegel befindet sich in seiner maximal oberen Position. Dies wird durch das Leuchten der LED signalisiert.
Action	Der Spiegel wird nach unten bewegt.
Expected Result	Die LED hoert auf zu leuchten.

Table A.101.: System Test Case LED_EM 12

Step	*Content*
Precondition	Der Spiegel befindet sich in seiner maximal unteren Position. Dies wird durch das Leuchten der LED signalisiert.
Action	Der Spiegel wird nach links bewegt.
Expected Result	Die LED hoert auf zu leuchten.

Test Cases for LED Finger Protection

Table A.102.: System Test Case LED_FP 1

Step	*Content*
Precondition	Das Fenster wird geschlossen. Die LED leuchtet nicht.
Action	Der Einklemmschutz wird aktiviert.
Expected Result	Die LED beginnt zu leuchten.

Table A.103.: System Test Case LED_FP 1

Step	*Content*
Precondition	Das Fenster wird geschlossen. Die LED leuchtet nicht.
Action	Der Einklemmschutz wird nicht aktiviert.
Expected Result	Die LED beginnt nicht zu leuchten.

Table A.104.: System Test Case LED_FP 1

Step	*Content*
Precondition	Der Einklemmschutz ist aktiviert. Die LED leuchtet.
Action	Der Einklemmschutz wird deaktiviert.
Expected Result	Die LED hoert auf zu leuchten.

Test Cases for LED Exterior Mirror with Heating

Table A.105.: System Test Case LED_EMH 1

Step	*Content*
Precondition	Die Aussenspiegelheizung ist inaktiv.
Action	Die Aussenspiegelheizung wird aktiviert.
Expected Result	Die LED beginnt zu leuchten.

Table A.106.: System Test Case LED_EMH 2

Step	*Content*
Precondition	Die Aussenspiegelheizung ist inaktiv.
Action	Die Aussenspiegelheizung wird nicht aktiviert.
Expected Result	Die LED beginnt nicht zu leuchten.

Table A.107.: System Test Case LED_EMH 3

Step	*Content*
Precondition	Die Aussenspiegelheizung ist aktiv. Die LED leuchtet.
Action	Die Aussenspiegelheizung wird deaktiviert.
Expected Result	Die LED hoert auf zu leuchten.

Test Cases for LED Power Window

Note that the test cases for the LED power window are applicable for both types of power windows as long as the power window LED is installed.

Table A.108.: System Test Case LED_PW 1

Step	*Content*
Precondition	Die Fenster stehen still.
Action	Die Fenster werden nicht geoeffnet oder geschlossen.
Expected Result	Die LED leuchtet nicht.

Table A.109.: System Test Case LED_PW 2

Step	*Content*
Precondition	Die Fenster stehen still.
Action	Der Knopf zum Schliessen eines Fensters wird betaetigt. Das Fenster schliesst sich.
Expected Result	Die LED zeigt an, dass sich ein Fenster schliesst.

Table A.110.: System Test Case LED_PW 3

Step	*Content*
Precondition	Die Fenster stehen still.
Action	Der Knopf zum Oeffnen eines Fensters wird betaetigt. Das Fenster schliesst sich.
Expected Result	Die LED zeigt an, dass sich ein Fenster oeffnet.

Table A.111.: System Test Case LED_PW 4

Step	*Content*
Precondition	Ein Fenster schliesst sich.
Action	Das Schliessen des Fensters wird angehalten.
Expected Result	Die LED hoert auf zu leuchten.

Table A.112.: System Test Case LED_PW 5

Step	*Content*
Precondition	Ein Fenster oeffnet sich.
Action	Das Oeffnen des Fensters wird angehalten.
Expected Result	Die LED hoert auf zu leuchten.

Test Cases for Manual Power Window

Table A.113.: System Test Case ManPW 1

Step	*Content*
Precondition	Die Fenster stehen still.
Action	Der Knopf zum Schliessen eines Fensters wird betaetigt und gehalten.
Expected Result	Das Fenster schliesst sich.

Table A.114.: System Test Case ManPW 2

Step	*Content*
Precondition	Ein Fenster wird geschlossen.
Action	Das Druecken auf die Taste zum Schliessen des Fensters wird beendet.
Expected Result	Das Fenster haelt an.

Table A.115.: System Test Case ManPW 3

Step	*Content*
Precondition	Die Fenster stehen still.
Action	Der Knopf zum Oeffnen eines Fensters wird betaetigt und gehalten.
Expected Result	Das Fenster oeffnet sich.

Table A.116.: System Test Case ManPW 4

Step	*Content*
Precondition	Ein Fenster wird geoeffnet.
Action	Das Druecken auf die Taste zum Oeffnen des Fensters wird beendet.
Expected Result	Das Fenster haelt an.

Table A.117.: System Test Case ManPW 5

Step	*Content*
Precondition	Ein Fenster wird geschlossen.
Action	Das Fenster wird komplett geschlossen.
Expected Result	Das Fenster haelt an.

Table A.118.: System Test Case ManPW 6

Step	*Content*
Precondition	Ein Fenster wird geoeffnet.
Action	Das Fenster wird komplett geoeffnet.
Expected Result	Das Fenster haelt an.

Table A.119.: System Test Case ManPW 7

Step	*Content*
Precondition	Ein Fenster ist komplett geschlossen.
Action	Der Knopf zum Schliessen des Fensters wird betaetigt.
Expected Result	Das Fenster bleibt geschlossen.

Table A.120.: System Test Case ManPW 8

Step	*Content*
Precondition	Ein Fenster ist komplett geoeffnet.
Action	Der Knopf zum Oeffnen des Fensters wird betaetigt.
Expected Result	Das Fenster bleibt geoeffnet.

Table A.121.: System Test Case ManPW 9

Step	*Content*
Precondition	Das Fenster ist nicht komplett geschlossen.
Action	Die Taste zum Schliessen wird angetippt.
Expected Result	Das Fenster beginnt sich zu schliessen und bleibt stehen, sobald der Druck nachlaesst.

Table A.122.: System Test Case ManPW 10

Step	*Content*
Precondition	Das Fenster ist nicht komplett geoeffnet.
Action	Die Taste zum Oeffnen wird angetippt.
Expected Result	Das Fenster beginnt sich zu oeffnen und bleibt stehen, sobald der Druck nachlaesst.

Test Cases for Remote Control Key

Table A.123.: System Test Case RCK 1

Step	*Content*
Precondition	Die Zentralverriegelung ist inaktiv.
Action	Der Fernbedienungsknopf zum Verriegeln wird betaetigt.
Expected Result	Die Zentralverriegelung wird aktiviert.

Table A.124.: System Test Case RCK 2

Step	*Content*
Precondition	Die Zentralverriegelung ist aktiv.
Action	Der Fernbedienungsknopf zum Entriegeln wird betaetigt.
Expected Result	Die Zentralverriegelung wird deaktiviert.

Table A.125.: System Test Case RCK 3

Step	*Content*
Precondition	Die Zentralverriegelung ist aktiv.
Action	Der Fernbedienungsknopf zum Verriegeln wird betaetigt.
Expected Result	Die Zentralverriegelung bleibt aktiviert.

Table A.126.: System Test Case RCK 4

Step	*Content*
Precondition	Die Zentralverriegelung ist inaktiv.
Action	Der Fernbedienungsknopf zum Entriegeln wird betaetigt.
Expected Result	Die Zentralverriegelung bleibt deaktiviert.

Test Cases for Safety Function

Table A.127.: System Test Case SF 1

Step	*Content*
Precondition	Eine Tür wurde entriegelt.
Action	Die Tür wird innerhalb von 10 Sekunden nach Entriegelung nicht geoeffnet.
Expected Result	Die Tür wird wieder verriegelt.

Table A.128.: System Test Case SF 2

Step	*Content*
Precondition	Eine Tür wurde entriegelt.
Action	Die Tür wird innerhalb von 10 Sekunden nach Entriegelung geoeffnet.
Expected Result	Die Tür wird nicht wieder verriegelt.

Technische Universität Braunschweig
Informatik-Berichte ab Nr. 2010-01

2010-01	A. Litvinenko and H. G. Matthies	Sparse data formats and efficient numerical methods for uncertainties quantification in numerical aerodynamics
2010-02	D. Grunwald, M. Lochau, E. Börger, U. Goltz	An Abstract State Machine Model for the Generic Java Type System
2010-03	M. Krosche, R. Niekamp	Low-Rank Approximation in Spectral Stochastic Finite Element Method with Solution Space Adaption
2011-01	L. Märtin, M. Schatalov, C. Knieke	Entwicklung und Erweiterung einer Werkzeugkette im Kontext von IT-Ökosystemen
2011-02	B. V. Rosić, A. Litvinenko, O. Pajonk, H. G. Matthies	Direct Bayesian update of polynomial chaos representations
2011-03	H. G. Matthies	White Noise Analysis for Stochastic Partial Differential Equations
2011-04	O. Pajonk, B.V. Rosić, A. Litvinenko, and H. G. Matthies	A Deterministic Filter for non-Gaussian Bayesian Estimation
2011-05	H. G. Matthies	A Hitchhiker's Guide to Mathematical Notation and Definitions
2011-06	R. van Glabbeek, U. Goltz, J.-W. Schicke	On Causal Semantics of Petri Nets
2011-07	H. Cichos, S. Oster, M. Lochau, A. Schürr	Extended Version of Model-based Coverage-Driven Test Suite Generation for Software Product Lines
2011-08	W.-B. Pöttner, J. Morgenroth, S. Schildt, L. Wolf	An Empirical Performance Comparison of DTN Bundle Protocol Implementations
2011-09	H. G. Matthies, A. Litvinenko, O. Pajonk, B. V. Rosić and E. Zander	Parametric and Uncertenty Computations with Tensor Product Representations
2011-10	B. V. Rosić, A. Kučerová, J. Sýkora, A. Litvinenko, O. Pajonk and H. G. Matthies	Parameter Identification in a Probabilistic Setting
2011-11	M. Espig, W. Hackbusch, A. Litvinenko, H. G. Matthies and E. Zander	Efficient Analysis of High Dimensional Data in Tensor Formats
2011-12	S. Oster	A Semantic Preserving Feature Model to CSP Transformation
2012-01	O. Pajonk, B. V. Rosić and H. G. Matthies	Deterministic Linear Bayesian Updating of State and Model Parameters for a Chaotic Model
2012-02	B. V. Rosić and H. G. Matthies	Stochasticc Plasticity - A Variational Inequality Formulation and Functional Approximation Approach I: The Linear Case
2012-03	J. Rang	An analysis of the Prothero–Robinson example for constructing new DIRK and ROW methods
2012-04	S. Kolatzki, M. Hagner, U. Goltz and A. Rausch	A Formal Definition for the Description of Distributed Concurrent Components - Extended Version
2012-05	M. Espig, W. Hackbusch, A. Litvinenko, H. G. Matthies and P. Wähnert	Efficient low-rank approximation of the stochastic Galerkin matrix in tensor formats
2012-06	S. Mennicke	A Petri Net Semantics for the Join-Calculus
2012-07	S. Lity, R. Lachmann, M. Lochau, I. Schaefer	Delta-oriented Software Product Line Test Models - The Body Comfort System Case Study

Institute for Software Engineering and Automotive Informatics
Müehlenpfordtstr. 23
38106 Braunschweig