**Software Architecture**

***SteerTurnIllum***



|  |  |  |
| --- | --- | --- |
| Author: |  | Nicolae-Bogdan Bacrău |
| Version: |  | 1.0 |

Index

1 Glossary 3

2 Introduction 4

2.1 Purpose of the Document 4

2.2 Overview 4

3 Building Blocks 5

4 Virtual Functional Bus 7

5 Features 9

5.1 Vehicle Management 10

5.1.1 Components 10

5.1.2 Runtime 10

5.2 Steering 13

5.2.1 Components 13

5.2.2 Runtime 13

5.3 Turning 16

5.3.1 Components 16

5.3.2 Runtime 16

5.4 Illumination 19

5.4.1 Components 19

5.4.2 Runtime 19

6 Information about this Document 22

6.1 Copyright 22

6.2 Version Index 22

# Glossary

This section contains a glossary of all the important terms and acronyms used inside the document.

|  |  |
| --- | --- |
| **Term / Acronym** | **Description** |
| AUTOSAR | AUTomotive Open System ARchitecture |
| VFB | Virtual Functional Bus |
| SWC | Software Component |
| RTE | Runtime Environment |
| BSW | Basic Software |
| OS | Operating System |
| S/R | Sender / Receiver |
| C/S | Client / Server |
| ECU | Electronic Control Unit |
| uC | Microcontroller |
| ADC | Analog Digital Converter |
| DIO | Digital Input / Output |
| PWM | Pulse Width Modulation |

Table 1 - Glossary.

# Introduction

## Purpose of the Document

The purpose of the document is to define the software architecture of the ***SteerTurnIllum*** embedded academy project.

## Overview

The software architecture of ***SteerTurnIllum*** is based on AUTOSAR and consists of the following:

* **Building Blocks** – enumerates all the software system’s modules and their abstraction level.
* **Virtual Functional Bus** – defines the communication between all the software system’s SWCs through ports and connectors.
* **Features** – defines the components diagram and the runtime sequence diagrams for each feature.

# Building Blocks

1. The building blocks of the software system shall consist of the components illustrated in **Figure 1.**

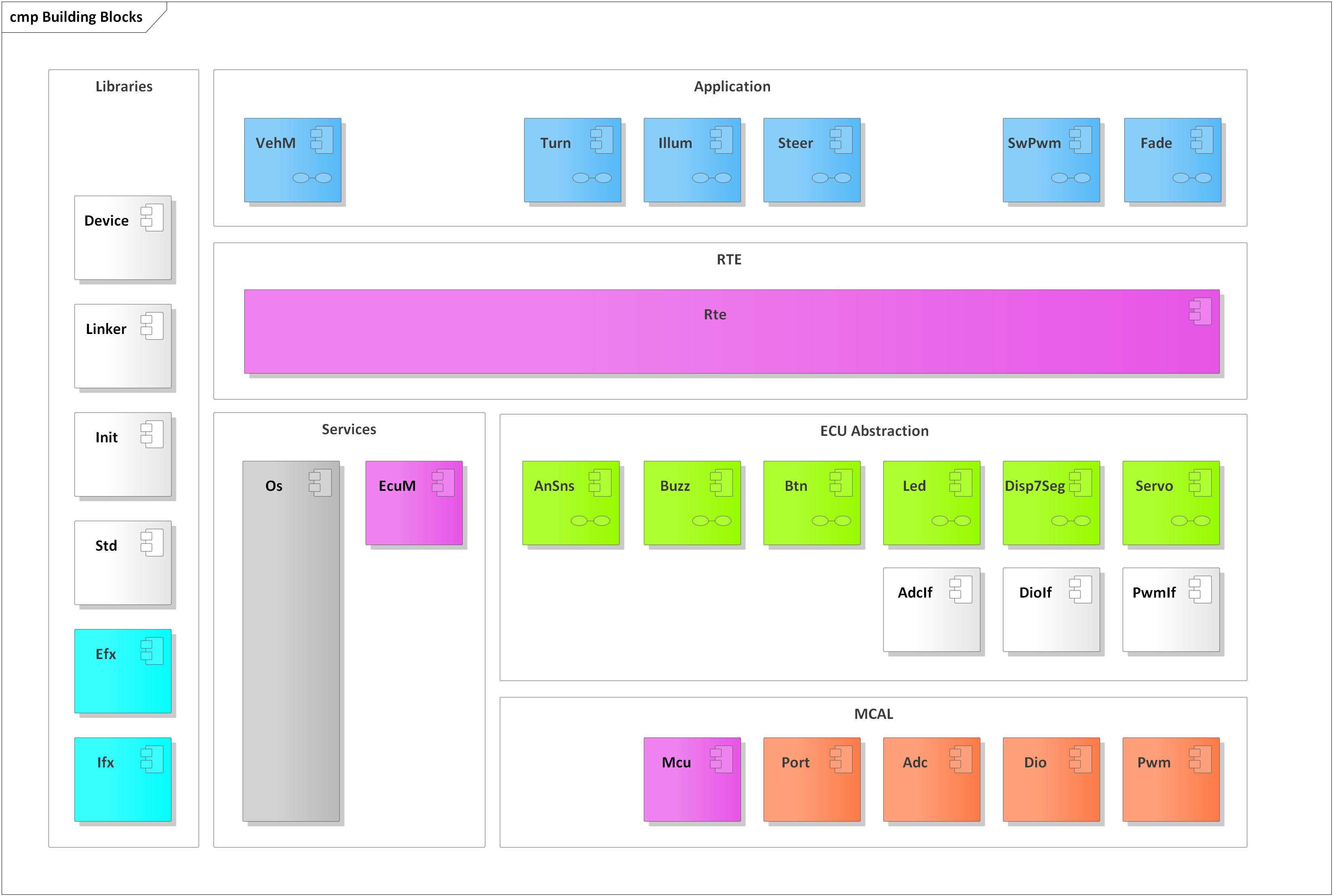


Figure 1 - Building Blocks Diagram.

The colors of the modules in the building blocks have the following significance:

* **White**: provided as full source code; no implementation required.
* **Grey**: provided as libraries and headers; no implementation required.
* **Purple**: full implementation provided as libraries; partial implementation provided as source code; requires partial implementation.
* **Cyan, Red, Green and Blue**: full implementation provided as libraries; requires full implementation.

Each of the **Purple**, **Cyan, Red, Green** and **Blue** modules are described in detail in a dedicated separate design requirements document.

The **White** and **Grey** modules are part of the framework and have the following purposes:

* **Device**:
  + Contains data types and masks for working with the uC registers.
* **Linker**:
  + Contains the description of linker script symbols that are needed for initializing RAM sections.
* **Init (Initialization)**:
  + Contains data types and functions for writing lists of (8, 16 or 32 bit) variables to desired initialization values.
  + The functions are of two types: full, which fully write the variables to the given values, or masked, which use AND and OR masks for writing only parts of the variables.
* **Std (Standard Definitions)**:
  + Contains the definition of the AUTOSAR data types and macros.
* **AdcIf (ADC Interface)**:
  + Contains data type and interface definitions for accessing ADCs on the ECU level. In this implementation only uC internal ADC channels are used, so the data types and interfaces are defined as the ones from the ADC driver.
  + Redefines the ADC channels from names containing the uC port name and port channel ID to names referencing the peripherals from the hardware schematic.
* **DioIf (DIO Interface)**:
  + Contains data type and interface definitions for accessing DIOs on the ECU level. In this implementation only uC internal DIO channels and ports are used, so the data types and interfaces are defined as the ones from the DIO driver.
  + Redefines the DIO channels and ports from names containing the uC port name and port channel ID to names referencing the peripherals from the hardware schematic.
* **PwmIf (PWM Interface)**:
  + Contains data type and interface definitions for accessing PWMs on the ECU level. In this implementation only uC internal PWM channels are used, so the data types and interfaces are defined as the ones from the PWM driver.
  + Redefines the PWM channels from names containing the uC port name and port channel ID to names referencing the peripherals from the hardware schematic.
* **Os (Operating System)**:
  + Makes use of a hardware timer for keeping track of the elapsed time and cyclically calls (with statically configured periodicities) OS tasks, implemented in the RTE.

# Virtual Functional Bus

1. The Virtual Functional Bus of the software system, defining the interaction between all SWCs, shall consist of the ports and connectors illustrated in **Figure 2**.

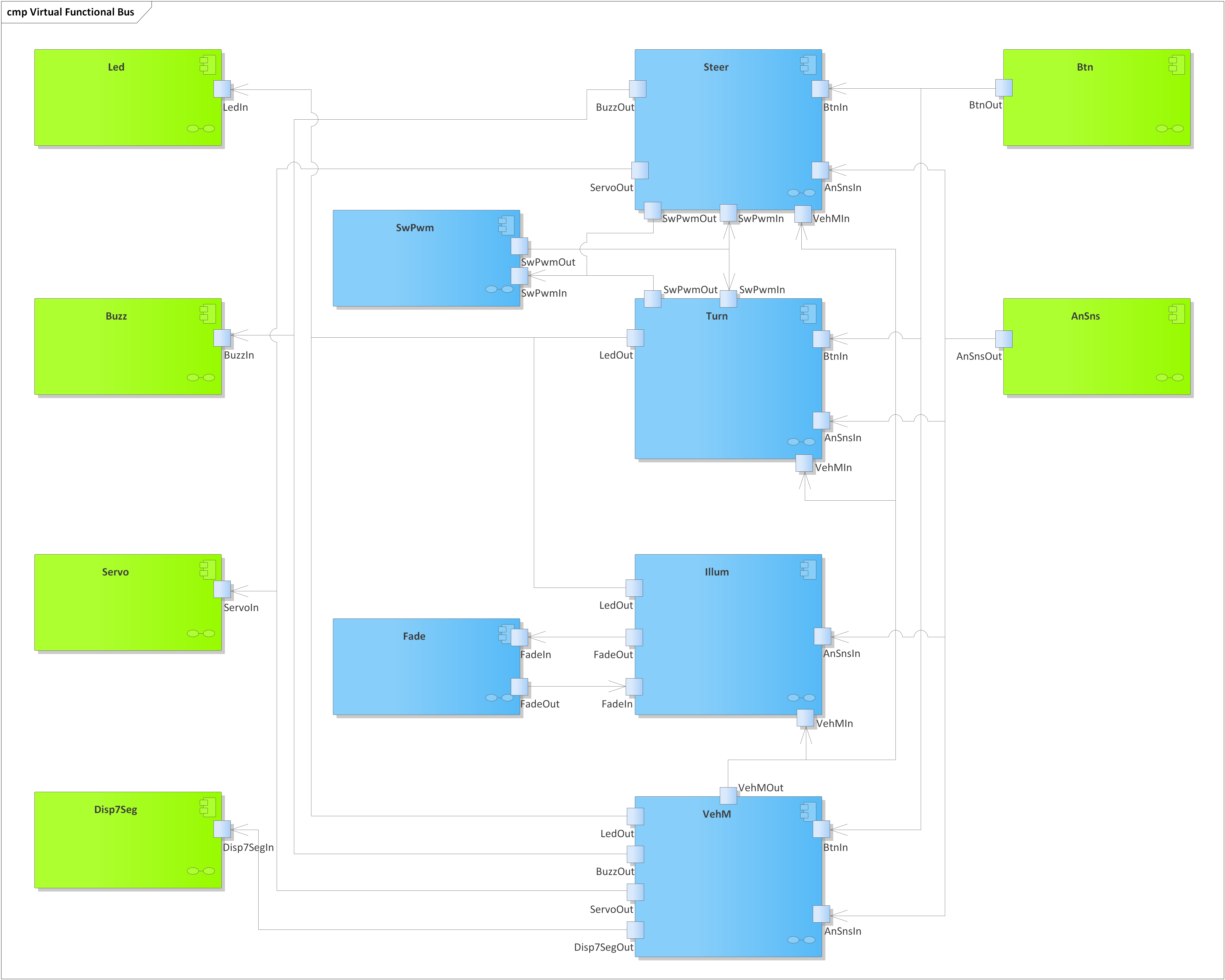


Figure 2 - Virtual Functional Bus diagram.

There are two types of SWCs:

* **Application SWCs (Blue)**:
  + Located above the RTE, in the Application layer.
  + Only use RTE interfaces for interacting with the BSW and other SWCs.
  + No hardware dependencies.
* **ECU Abstraction SWCs (Green)**:
  + Located below the RTE, in the ECU Abstraction layer.
  + Use RTE interfaces for interacting with BSW and the other SWCs.
  + Direct access to the ECU Abstraction interface modules for accessing the ADC, DIO and PWM resources on an ECU level.

The SWCs are software modules that implement a clearly defined behavior and can interact with each other only via the RTE, through clearly defined ports. In this architecture all the ports are S/R ports. No C/S ports are used.

The S/R ports are logical groups of interfaces for reading or writing data. The SWCs cannot call functions from other modules nor directly read / write global variables declared in other modules.

In order to assure the interaction between SWCs, the RTE declares global data buffers (global variables) and provides read and write interfaces for getting and setting their values. Each SWC has its own unique set of RTE read and write interfaces, accessible by including a dedicated header (*Rte\_<SWC\_Name>.h*).

Each SWC exports runnables. A runnable is a special type of function that can only be called inside an RTE task. There are two types of runnables:

* **Initialization runnable** – called only once, after reset, in the initialization task.
* **Cyclic runnables** (also called main functions) – called inside cyclic RTE tasks; which task depends on the timing constraints of the system requirements (it is up to the developer to choose).

# Features

The software features in this chapter are correlated with the system features from the system requirements. Each software feature defines:

* **Components diagram**: defines the list of software components and the interaction between them through the interfaces that are relevant for the given feature.
* **Runtime sequence diagram:** presents the order of execution of all the runnables and the interactions with the RTE through S/R interfaces, defining the chronological data flow between the components.

For simplicity, some interfaces, although are part of the components, do not appear in some feature diagrams since they are not relevant for that feature.

The components and runtime sequence diagrams illustrate the actual S/R interfaces (C functions) that are used for communicating. The interfaces follow a naming convention:

* *Rte\_Read\_<PortName>\_<DataElementName>* – for reading data elements.
* *Rte\_Write\_<PortName>\_<DataElementName>* – for writing data elements.

The software features are executed in parallel.

## Vehicle Management

### Components

1. The Vehicle Management feature shall consist of the components, interfaces and dependencies illustrated in the components diagram from **Figure 3**.

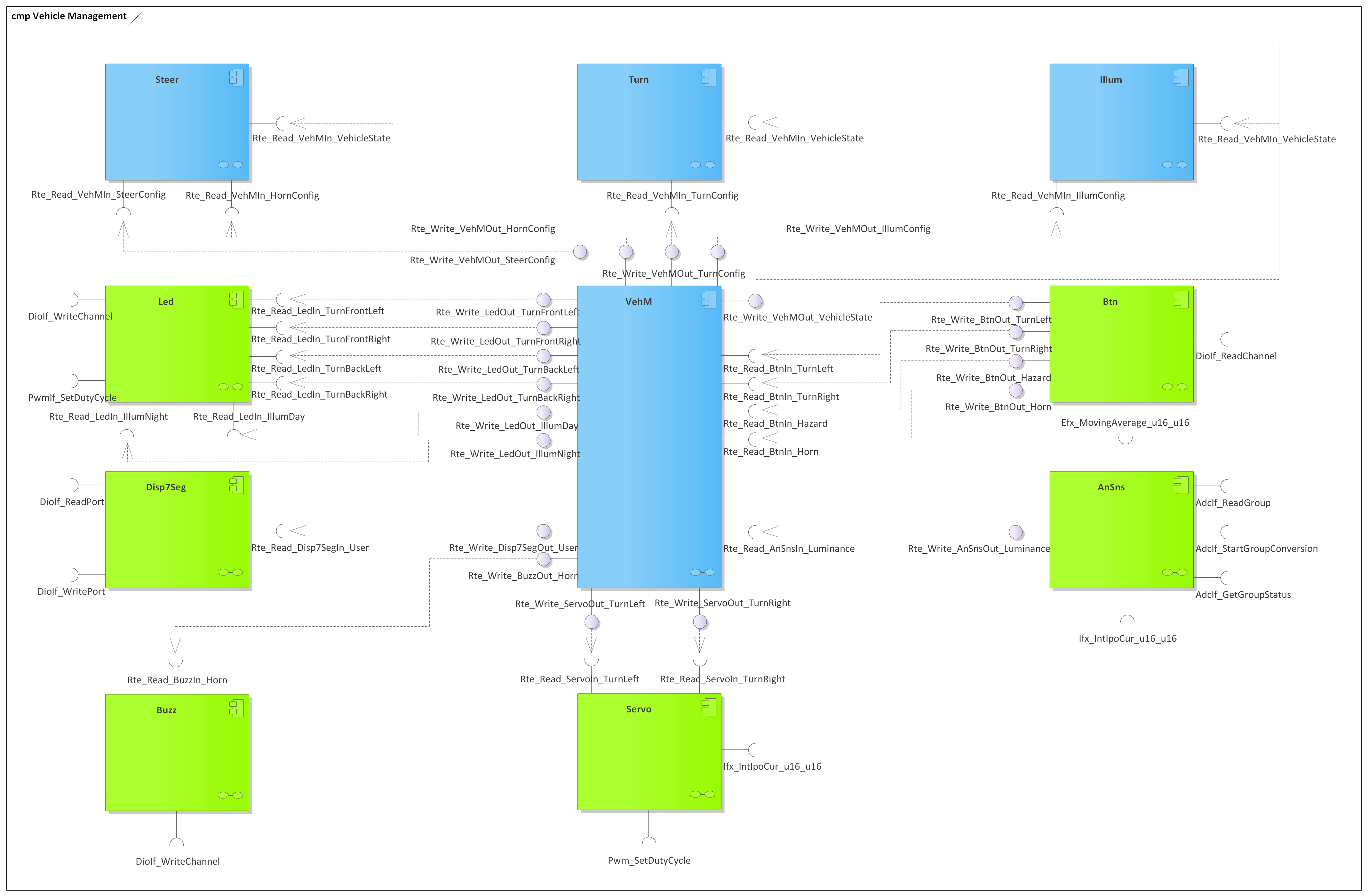


Figure 3 - Vehicle Management components diagram.

### Runtime

1. The Vehicle Management feature shall adhere to the execution order illustrated in the runtime sequence diagram from **Figure 4**.

**A picture containing graphical user interface

Description automatically generated**

A picture containing table

Description automatically generated

Figure 4 - Vehicle Management runtime sequence diagram.

## Steering

### Components

1. The Steering feature shall consist of the components, interfaces and dependencies illustrated in the components diagram from **Figure 5**.

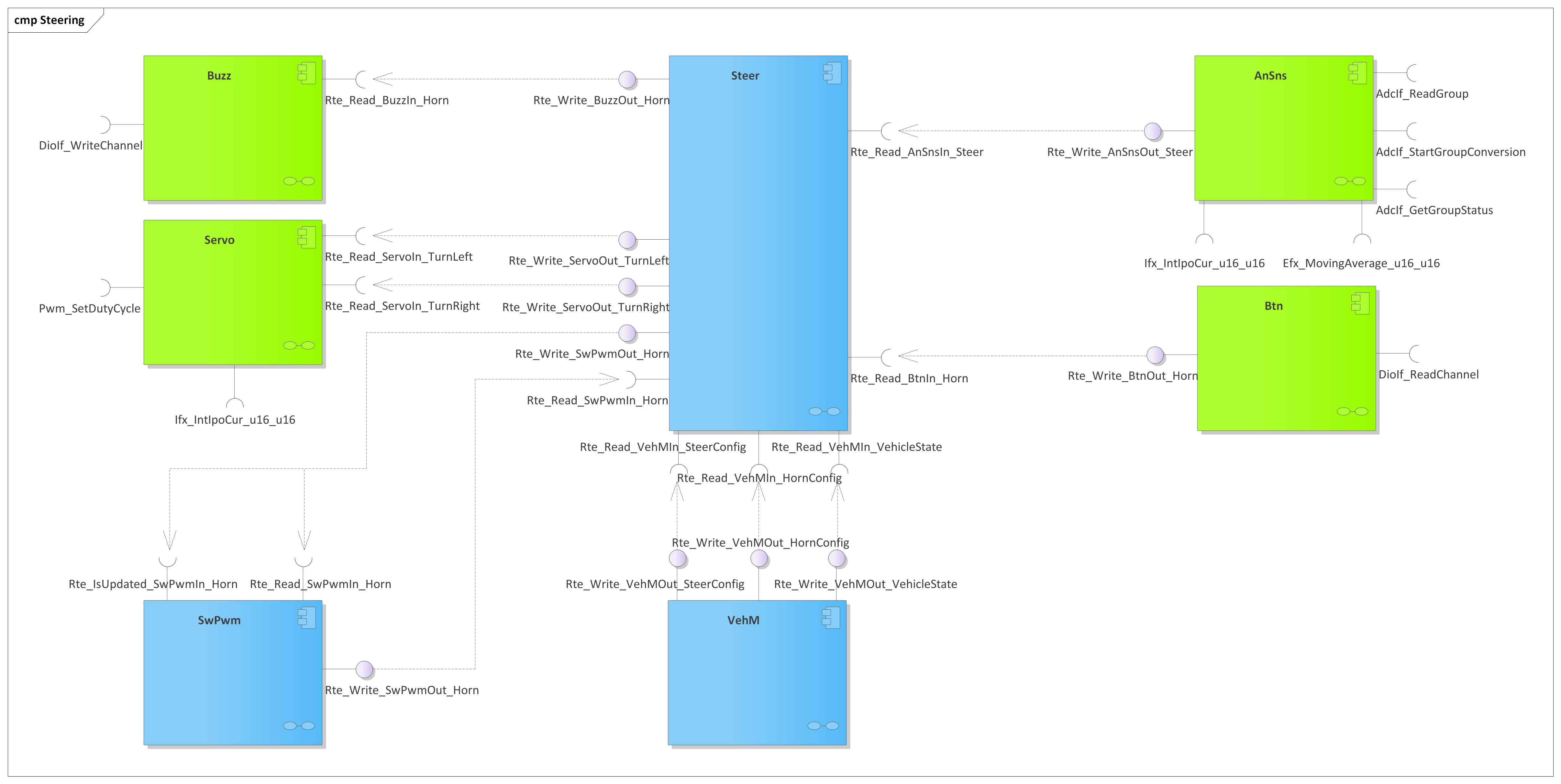


Figure 5 - Steering components diagram.

### Runtime

1. The Steering feature shall adhere to the execution order illustrated in the runtime sequence diagram from **Figure 6**.

A picture containing diagram

Description automatically generated

**Graphical user interface

Description automatically generated with low confidence**

Figure 6 - Steering runtime sequence diagram.

## Turning

### Components

1. The Turning feature shall consist of the components, interfaces and dependencies illustrated in the components diagram from **Figure 7**.



Figure 7 - Turning components diagram.

### Runtime

1. The Turning feature shall adhere to the execution order illustrated in the runtime sequence diagram from **Figure 8.**

Diagram

Description automatically generated

Table

Description automatically generated

Figure 8 - Turning runtime sequence diagram.

## Illumination

### Components

1. The Illumination feature shall consist of the components, interfaces and dependencies illustrated in the components diagram from **Figure 9**.

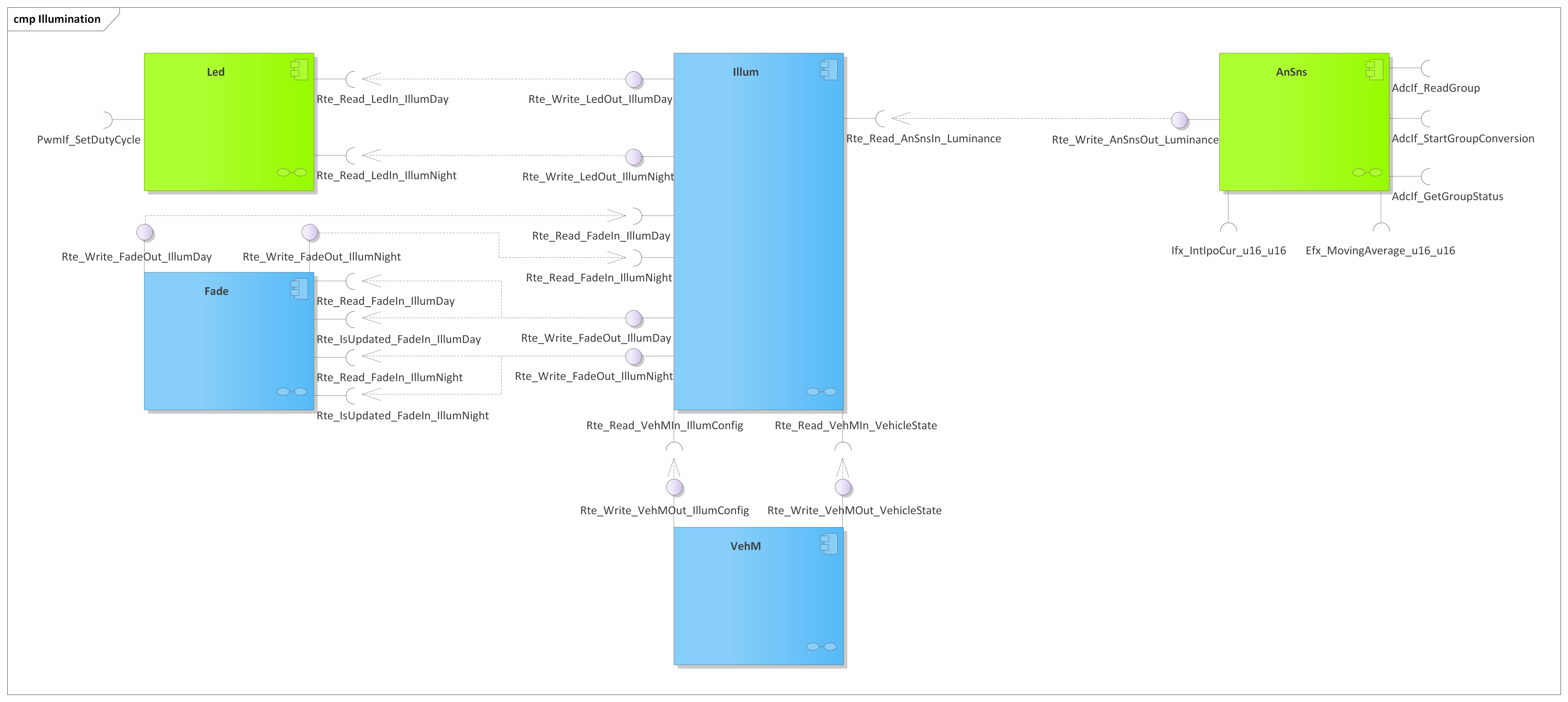
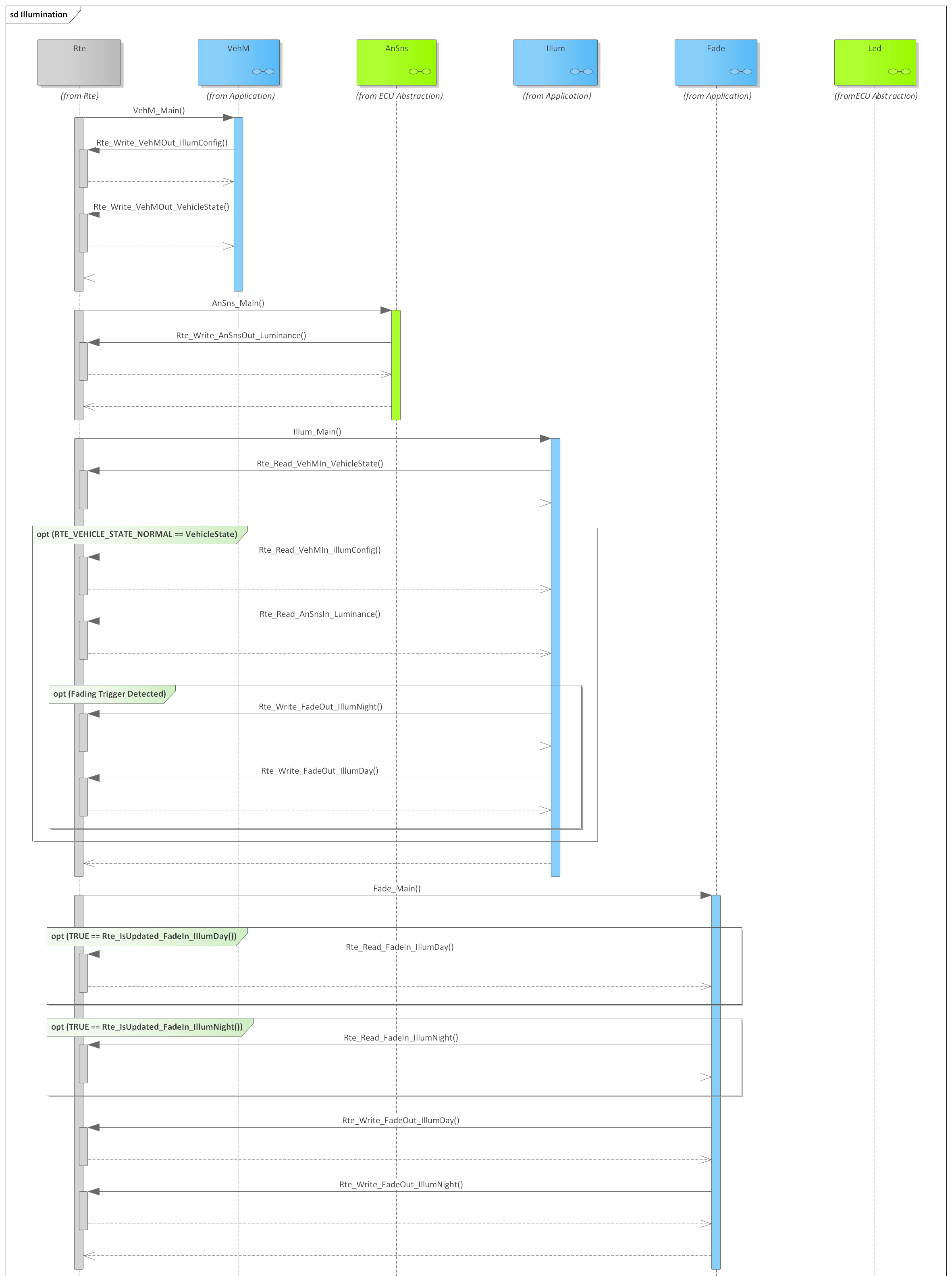


Figure 9 - Illumination components diagram.

### Runtime

1. The Illumination feature shall adhere to the execution order illustrated in the runtime sequence diagram from **Figure 10**.



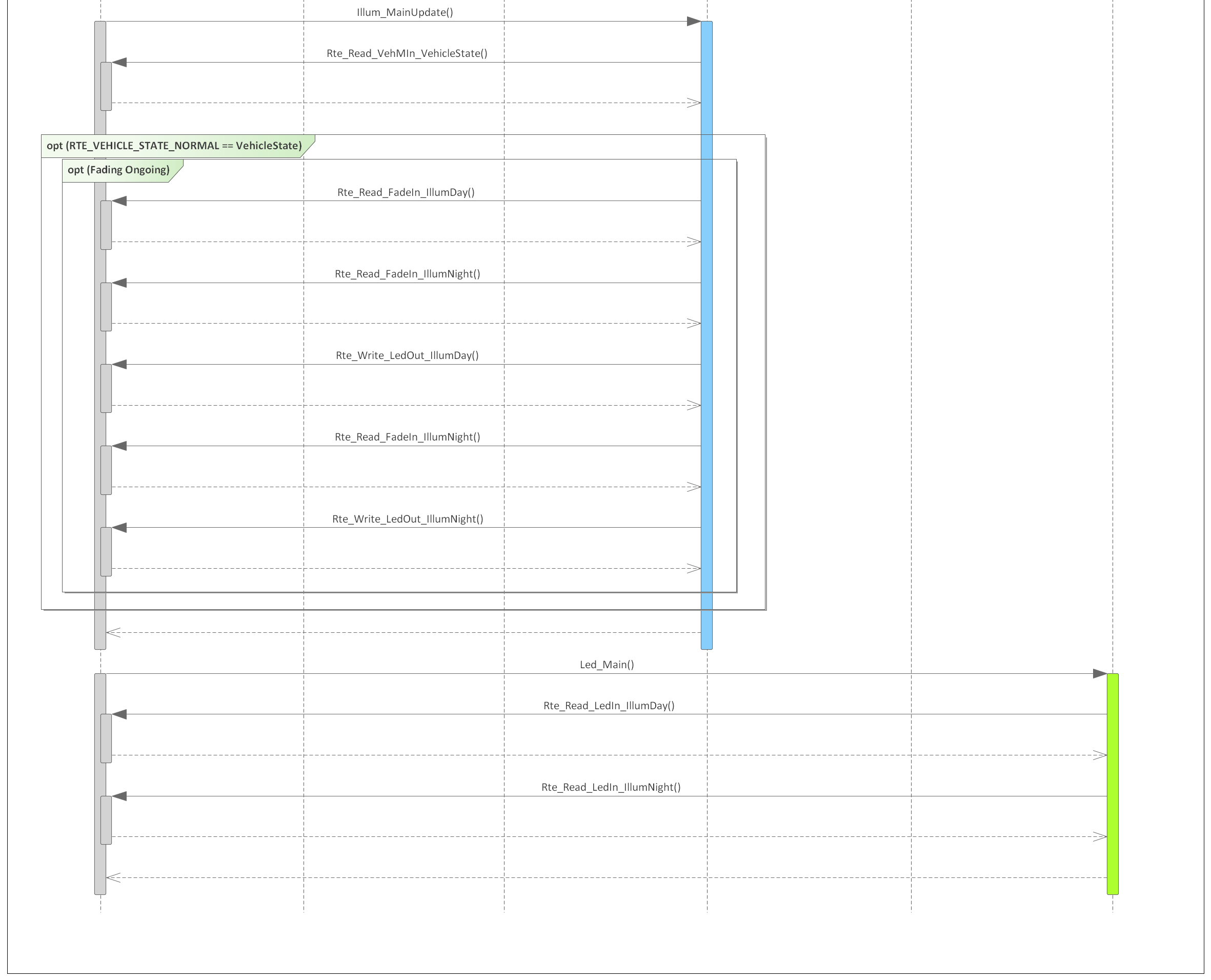


Figure 10 - Illumination runtime sequence diagram.

# Information about this Document

## Copyright

All rights, including translation rights, are reserved. Under no circumstances shall any fragment of this document be reproduced without written authorization from NTT DATA Romania S.A, including copying, photographing or replicating through other methods.

Copyright (c) NTT DATA Romania S.A.

## Version Index

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Chapter** | **Modification description** |
| 1.0 | 10.07.2021 | Nicolae-Bogdan Bacrău | All | Created. |

Table 2 - Version Index.