|  |  |
| --- | --- |
| **Title:** | **Car Lights**  **SW Component < XXXXXXX >** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **History** | | | | |
| **Issue status**  (Index) | **Maturity/Date**  (draft/invalid/valid)  (dd-mmm-yyyy) | **Author**  Department | **Check/Release**  Department | **Description** |
| 1.0 | DSD First Edition  27-August-09 | David Rosales / Diego Flores | David Rosales / Diego Flores | Creation of the document |

**Table of Contents**

**1** **PURPOSE**

**2** **DEFINITIONS AND ABBREVIATIONS**

**3** **REALIZATION CONSTRAINTS AND TARGETS**

**4** **SW COMPONENT INTERNAL BREAKDOWN**

**4.1** **Functional Decomposition**

[***4.2*** **Function *<Type> <function name> (type par 1, .., type par n)***](#h.35nkun2)

***4.3*** **Function <Type*> <function name> (type par 1, .., type par n)***

1. **Purpose**

*The purpose of this document is to design the software that will be in charge of controlling the car lights. There will be 3 possible Hardware configurations : Standard, High-End and Luxury.*

*Depending on the Hardware configuration, different light combinations will be used. Combination will include the Low beam lights, blinker lights, day lights and stop lights. All the possible combination are described in the body o this document.*

*CAN communication protocol will be implemented to control and report the status of the lights mentioned above, and all the command and parameter details are also described in this document.*

*The module of PWM will be used to control the power of the corresponding leds, as well as a non preemptive monotonic scheduler to control it real time.*

*A LDR sensor will be configured to measure the light and send signals to control the lights in the automatic mode.*

*All specification and details will be described in this document.*

1. **Definitions and abbreviations**

**Definitions**

|  |  |
| --- | --- |
| *Day-Stop lights*  *Day-Blinker lights* | *it refers to the combination of the stop lights and the back load day lights of the car.*  *it refers to the combination of the low beam lights and the front load day lights of the car* |
|  |  |
|  |  |
|  |  |
|  |  |

**Abbreviations**

*PELC Power Exterior Lighting Controller*

**References**

|  |  |  |
| --- | --- | --- |
| **N°** | **Document name** | **Reference** |
| *1*  *2* | *Traceability Matrix Template.xlsx*  *PFinal-BS\_PELC Requirements.doc* |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. **Realization constraints and targets**

*Represent the behavior of the exterior lights of a car emulating a module functions in charge to control these lights: PELC.*

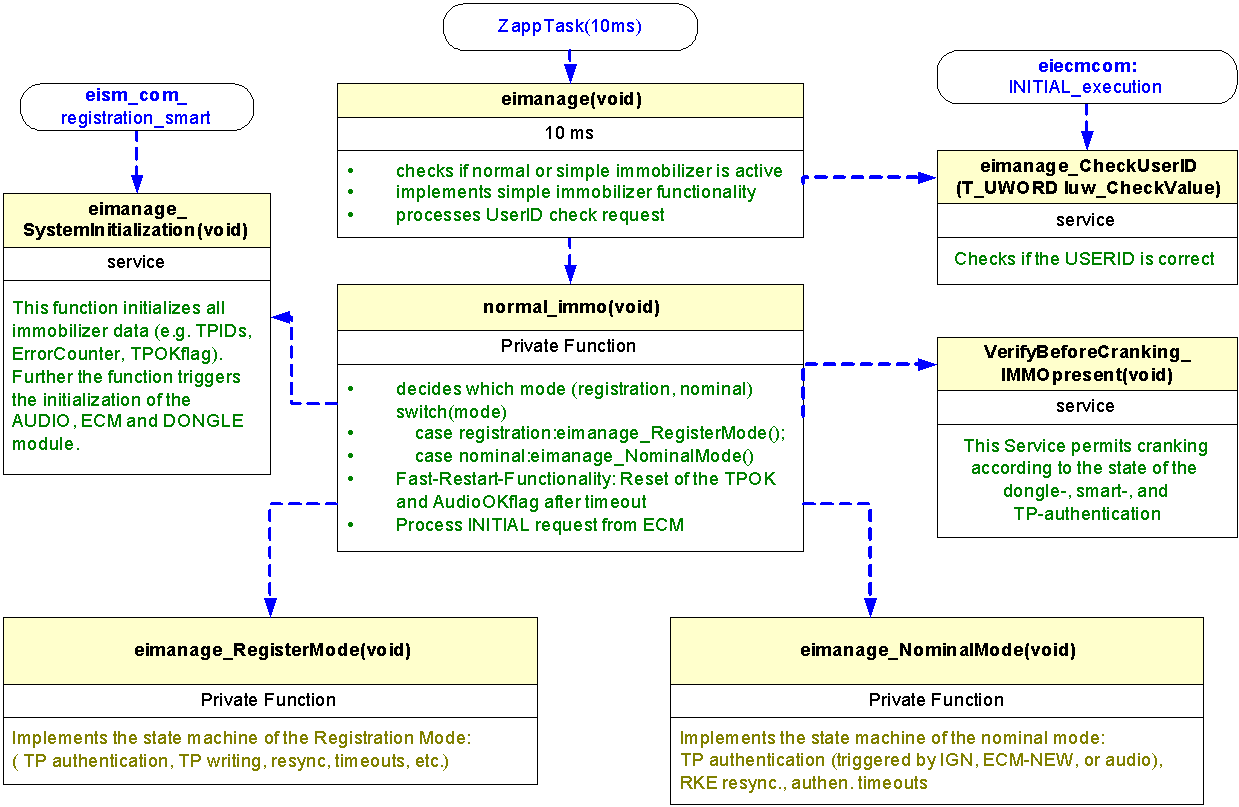
*this module must to obey the commands that will be send via CAN and ignore the commands that are not for this module*

*This module must be robust and portable capable of functioning in any kind of conditions.*

1. **SW Component internal breakdown**

*For complex SW Components, the designer may define SW Subcomponents. Please consider also the work step "Deal with complexity" provided by the method for Detailed SW Design.  
Note: SW Subcomponents are synonymous with the previous used term "Module"  
  
Mapping to the file structure:  
- Non complex SW Components should be represented by one object file.  
- For complex SW Components each SW Subcomponent should be represented by one object file.  
  
<Subcomponent decomposition if applicable>*

* 1. **Functional Decomposition**

*Overview of functions and their dependencies shown by a Static Function Tree*

**Function Description and Dynamic Behavior**

*Provide detailed static and dynamic description of all functions of the SW Component.   
Functions which are defined in other SW Components shall only be referenced in the external interface description!  
The signature description shall be done inside the function header in the source code.  
  
For each function, the following section should be copied*

* + 1. **Function *<Type> <function name> (type par 1, .., type par n)***

|  |  |
| --- | --- |
| **Description** | *Brief description of the function behavior and useful remarks* |
| **Parameter 1** <input| output| inout> | *Give an explanation if the parameter shall be checked by the user, or if a check is implemented in the function here* |
| **Parameter 2..n** |  |
| **Return Value** |  |
| **Precondition** | e.g. Function can only be called in a certain state, SW component is initialized Relation between input parameters where applicable (Input for Module Test) |
| **Post condition** | *e.g. specific State change e.g. car is locked, EEPROM Values written, Relation between output parameters where applicable* |
| **Error Conditions** |  |

**Dynamic Behavior**

State Chart1, Flow Chart1

1 *Preferred*  
*In this document, the dynamic behavior shall be designed on an abstract level showing the principle workflow of a function. Do not show the detailed implementation to ensure that the design description can be maintained with a reasonable effort. The target is not to show the complete detailed implementation 1:1.  
  
The detailed design shall reflect in detail what a function is doing from a black box view. The internal details are useful on an abstract, but not very detailed level.****If the function is not complex a short textual description might be sufficient and a graphical description is not needed.***

*Symbol and function names shall be self explaining.   
The link to the implementation may be provided by using the same names as in the design or by a comment showing the full name followed by the declaration showing the implementation.*

* + 1. **Function <Type*> <function name> (type par 1, .., type par n)***

|  |  |
| --- | --- |
| **Description** | Brief description of the function behavior and useful remarks |
| **Parameter 1** <input| output| inout> |  |
| **Parameter 2..n** |  |
| **Return Value** |  |
| **Precondition** | e.g. Function can only be called in a certain state, SW component is initialized Relation between input parameters where applicable (Input for Module Test) |
| **Post condition** | *e.g. specific State change e.g. car is locked, EEPROM Values written, Relation between output parameters where applicable* |
| **Error Conditions** |  |

**Dynamic Behavior**

State Chart1, Flow Chart1, Nassi Shneiderman

1 *Preferred*  
  
*In this document, the dynamic behavior shall be designed on an abstract level showing the principle workflow of a function. Do not show the detailed implementation to ensure that the design description can be maintained with a reasonable effort. The target is not to show the complete detailed implementation 1:1.  
  
The detailed design shall reflect in detail what a function is doing from a black box view. The internal details are useful on an abstract, but not on a very detailed level.   
  
Symbol and function names shall be self explanatory.   
The link to the implementation may be provided by using the same name as in the design or by a comment showing the full name followed by the declaration showing the implementation.*

* 1. **Macros**

*Only complex macros shall be described here. A complex macro is longer then one line or has more then one parameter or command. A complete description of all macros shall be done by code comments.  
Be aware: macros shouldn’t be part of the interface: Don’t use macros at interface level!  
If you still think you need macros: document them like functions.  
Internal Interfaces  
For internal interaction a data flow and/or a message sequence chart shall be provided. If the Static Function Tree already contains a data flow description then just reference it*

* 1. **SW module global variables**

*Where shared variables can not be avoided, a risk analysis is mandatory.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Memory section** | **Description** | **Shared** |
|
| Klstrg\_status | T\_UBYTE | NON\_INIT\_RAM | Status of terminal control | X |
|
|  |  |  |  |  |
|
|  |  |  |  |  |
|
|  |  |  |  |  |
|
|  |  |  |  |  |
|
|  |  |  |  |  |
|

* 1. **Design solutions for general features**

*It should describe detailed design solutions for general features of our products. i.e ”Layer Programming”*

* 1. **SW Component integration**
     1. **Integration context / constraints**

*Tbd: help text*

* + 1. **Include files**

*List here the necessary include files and their order,*

* + 1. **Initialization**

*e.g. Sequence Chart for initialization*  
All function bellow shall be called on initialization

|  |  |  |
| --- | --- | --- |
| Call Order | Service | Call constraints linked to an other module |
| *1* | *BCM\_Warning\_Output\_Init\_Function* | *After EEPROM init* |

* + 1. **Interrupts**

*If described in the SW Architecture Document, just put the reference here.*

* + 1. **Real time scheduler task**

*If described in the SW Architecture Document, just put the reference here.*

* + 1. **Other entry points**

|  |  |  |
| --- | --- | --- |
| **Name** | **Service** | **Task level** |
|  |  |  |