|  |  |
| --- | --- |
| **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-15 | David Rosales / Diego Flores | David Rosales / Diego Flores | Creation of the document |
| 1.0 | DSD First Edition  01-Sep-15 | David Rosales /  Diego Flores |  | Addition of diagrams |

**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*

*SM State Machine*

**References**

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

1. **Realization constraints and targets**

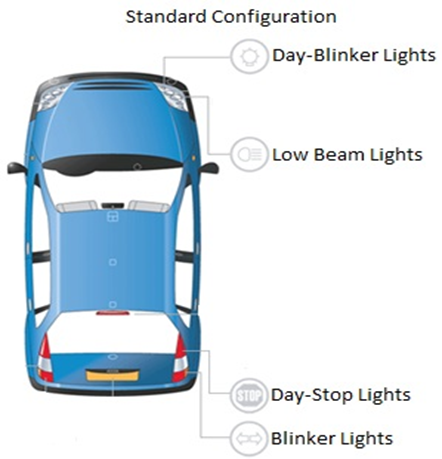
*Represent the behavior of the exterior lights of a car emulating a module functions in charge to control these lights: PELC.*

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

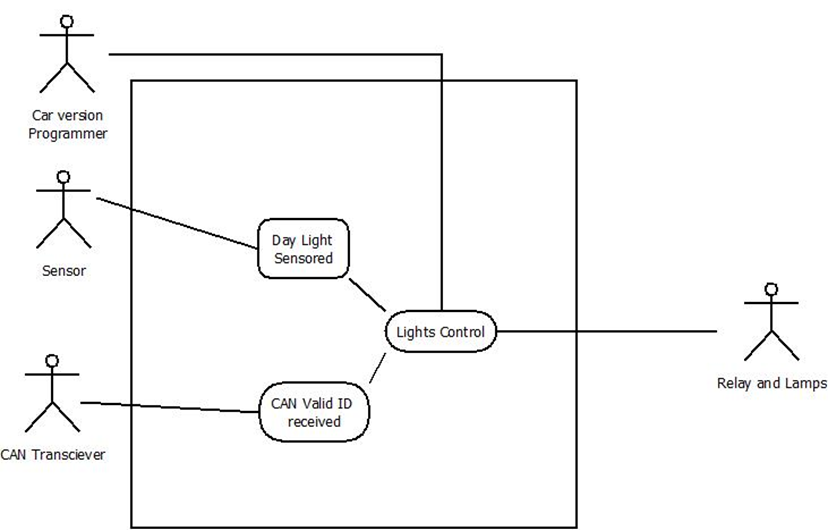
*This module must be robust and portable capable of functioning in any kind of conditions.*

1. **SW Component internal breakdown**

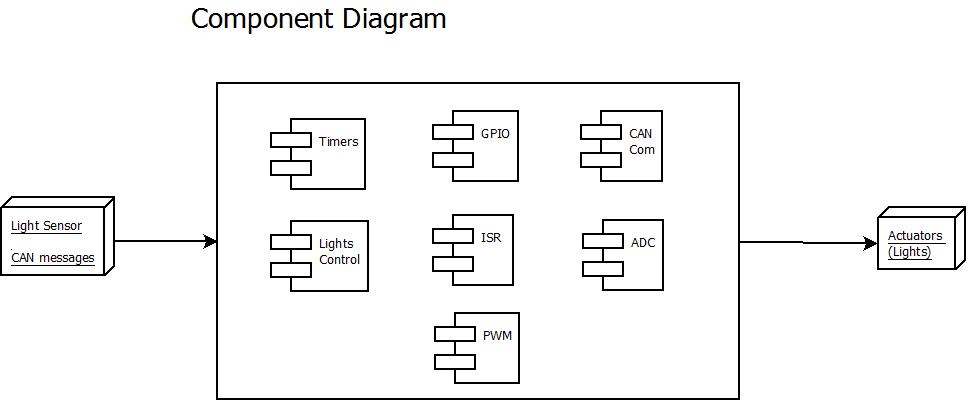
**Car Lights map**

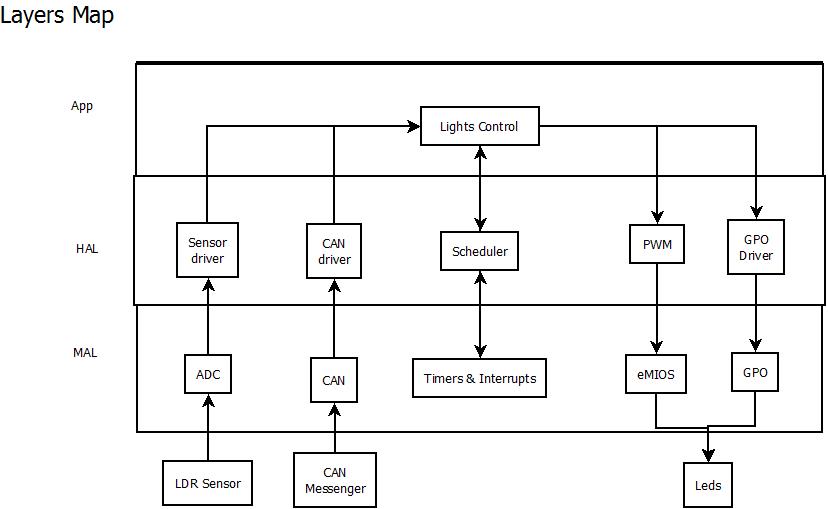
**

**Use Case Diagram**

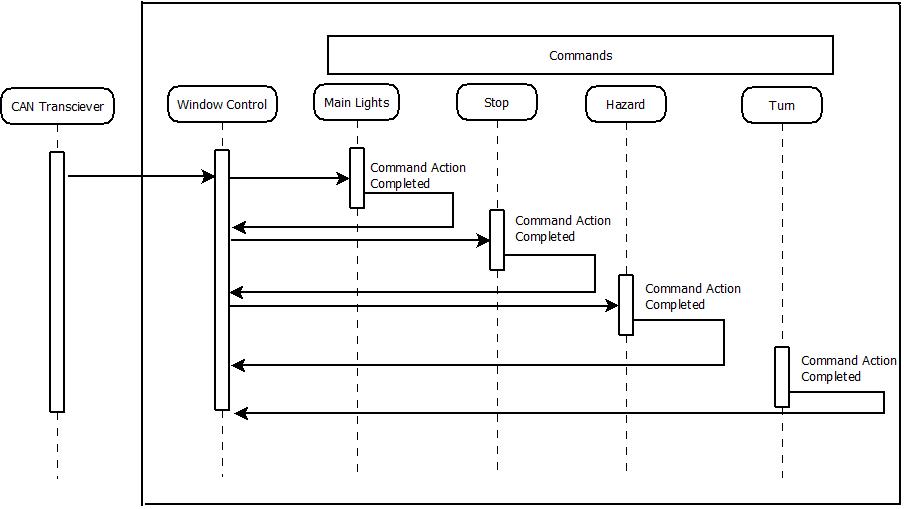


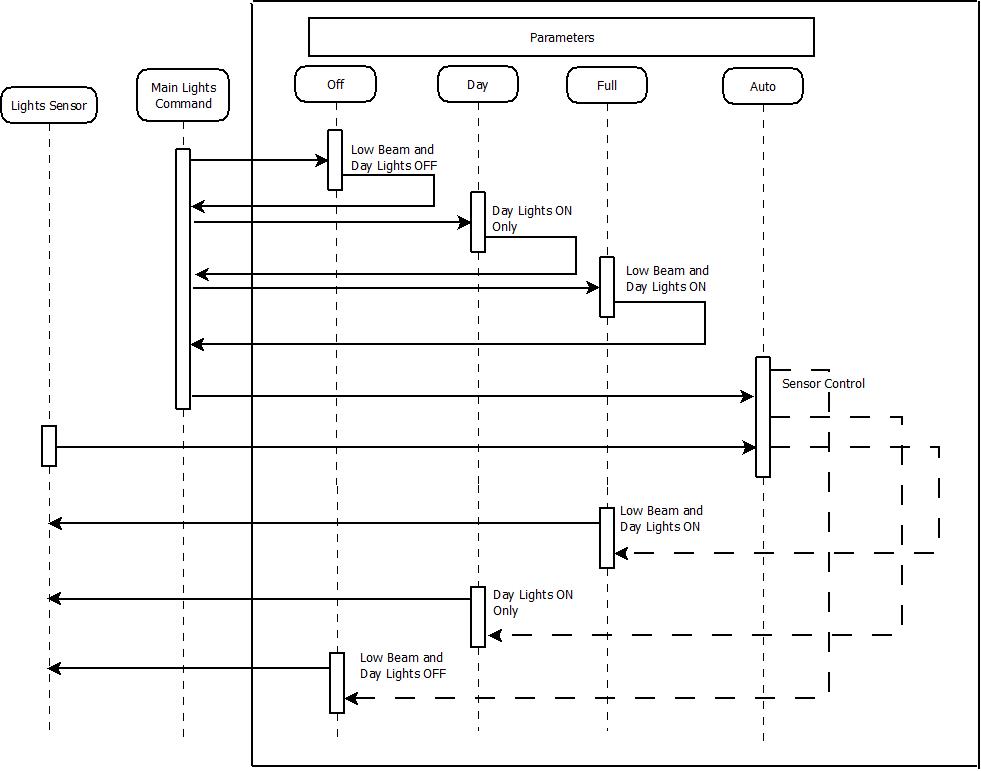
**Component Diagram**

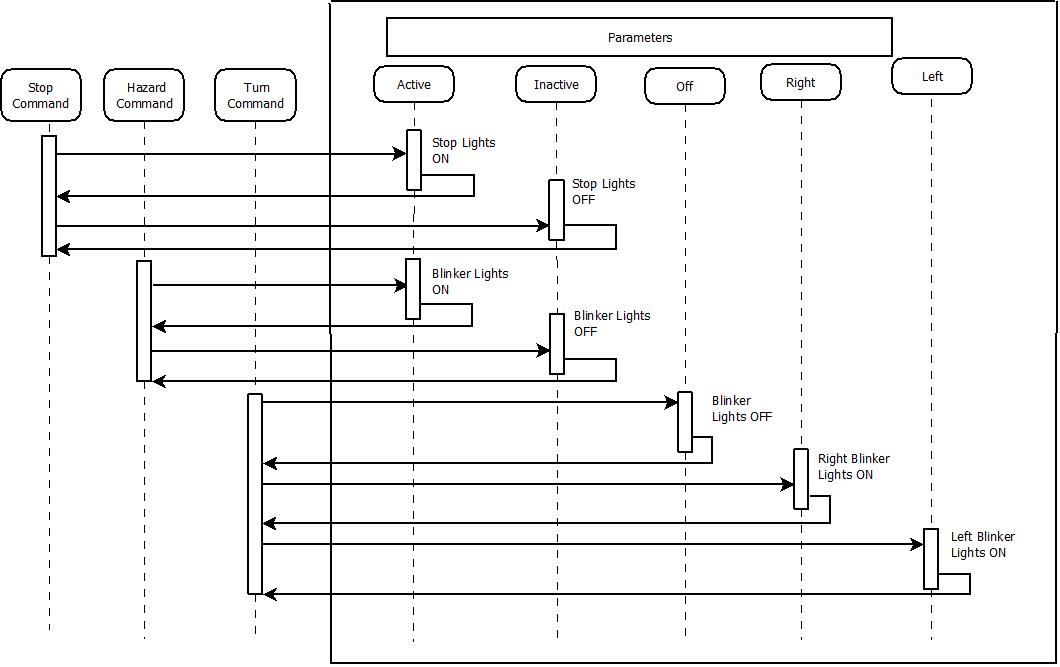




**Sequence Diagrams**

****

****

****

* 1. **Functional Decomposition**

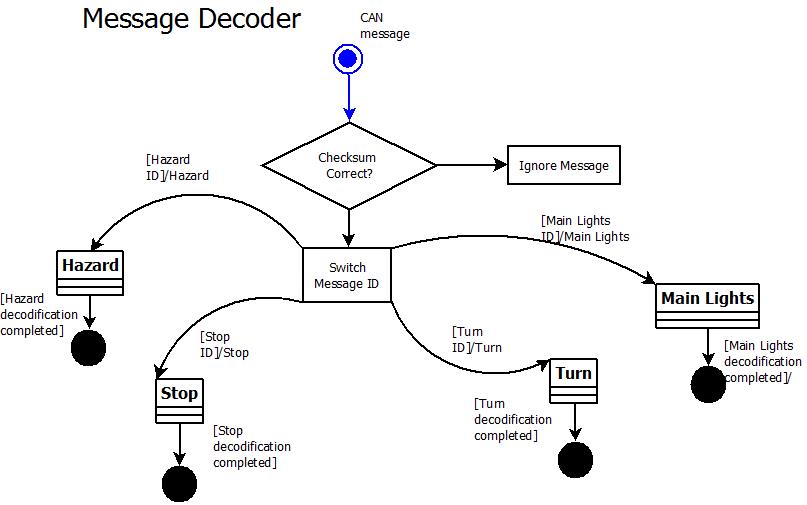
**Function Description and Dynamic Behavior**

*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 *void Message\_decoder(void)***

|  |  |
| --- | --- |
| **Description** | *This function receives the data contained in the CAN message, and decodes the complete message including the checksum calculation. It contains a state machines that refines and separates the ID and its corresponding parameters* |
| **Parameter 1** <input| output| inout> | *None* |
| **Parameter 2..n** |  |
| **Return Value** | None |
| **Precondition** | Function can be only called when a CAN message is received and an interrupt has taken place. |
| **Post condition** | *The state machine changes the parameter variables, and saves the data , so the data can be read by the lights control module* |
| **Error Conditions** |  |

**Dynamic Behavior**

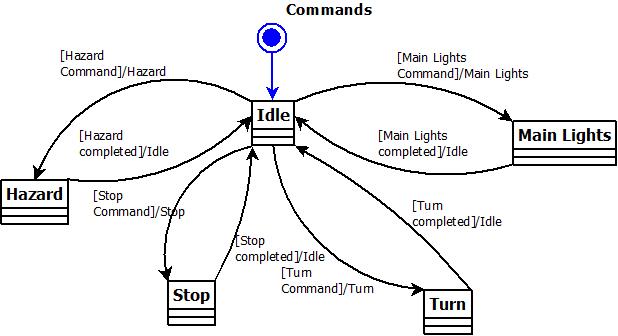


* + 1. **Function void lights\_control\_SM (void)**

|  |  |
| --- | --- |
| **Description** | This functions contains the main state machine for the lights control. it switched the value of the message ID and the default state is IDLE, where the message ID is read. No other logical functions are run. |
| **Parameter 1** <input| output| inout> | void |
| **Parameter 2..n** |  |
| **Return Value** | void |
| **Precondition** | This function is called by the scheduler every 25 ms, this means that our function is checking every 25ms if there is any valid mesage. |
| **Post condition** | *If the switched value changes, this state machine calls different function according to the command received.* |
| **Error Conditions** |  |

**Dynamic Behavior**

State Chart1, Flow Chart1, Nassi Shneiderman



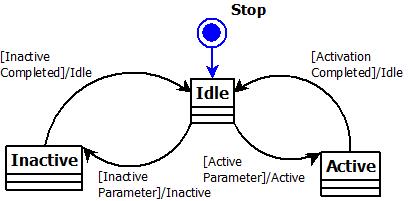
* + 1. **Function  *void Light\_Ctrl\_HwConfig(void)***

|  |  |
| --- | --- |
| **Description** | *This function sets the value of the car version selection. This value is taken from the Hardware selector, so the macros are defined based on the GPI* |
| **Parameter 1** <input| output| inout> | *None* |
| **Parameter 2..n** |  |
| **Return Value** | None |
| **Precondition** | Function can be only called at the begginin of the program, and it is called by the main function. |
| **Post condition** | *Once the version is set, the function finishes and the program continues with the scheduler tasks.* |
| **Error Conditions** |  |

**Dynamic Behavior**

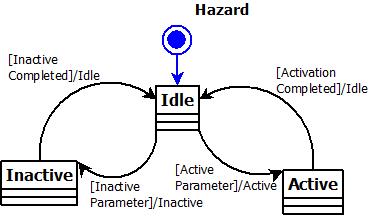
* + 1. **Function  *T\_UBYTE Command\_Stop (void)***

|  |  |
| --- | --- |
| **Description** | *This function contains the logic selection of all possible Stop commands, active or inactive, and sends instructions to GPO modules.* |
| **Parameter 1** <input| output| inout> | *None* |
| **Parameter 2..n** |  |
| **Return Value** | Variable IDLE is returned to the light control state machine, so we can read a different can message. |
| **Precondition** | Function can be only called by the light control state machine |
| **Post condition** | *Once the light output signals have been sent to GPO, this functions sends back IDLE to the state machine and finishes the stop process.* |
| **Error Conditions** |  |



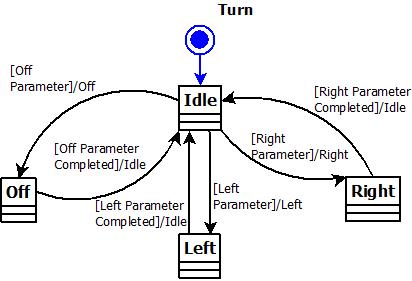
* + 1. **Function  *T\_UBYTE Command\_Hazard (void)***

|  |  |
| --- | --- |
| **Description** | *This function contains the logic selection of all possible Hazard commands, active or inactive, and the ON time and OFF time. it also sends instructions to GPO modules.* |
| **Parameter 1** <input| output| inout> | *None* |
| **Parameter 2..n** |  |
| **Return Value** | Variable IDLE is returned to the light control state machine, so we can read a different can message. |
| **Precondition** | Function can be only called by the light control state machine |
| **Post condition** | *Once the light output signals have been sent to GPO, this functions sends back IDLE to the state machine and finishes the Hazard process.* |
| **Error Conditions** |  |



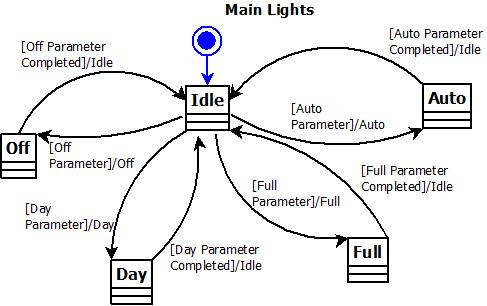
* + 1. **Function  *T\_UBYTE Command\_Turn (void)***

|  |  |
| --- | --- |
| **Description** | *This function contains the logic selection of all possible Turn commands, right, left or off., and sends instructions to GPO modules.* |
| **Parameter 1** <input| output| inout> | *None* |
| **Parameter 2..n** |  |
| **Return Value** | Variable IDLE is returned to the light control state machine, so we can read a different can message. |
| **Precondition** | Function can be only called by the light control state machine |
| **Post condition** | *Once the light output signals have been sent to GPO, this functions sends back IDLE to the state machine and finishes the turn process.* |
| **Error Conditions** |  |



* + 1. **Function *T\_UBYTE Command\_Mainlight (void)***

|  |  |
| --- | --- |
| **Description** | *This function contains the logic selection of all possible Mainlights commands, day, full, off or auto, and sends instructions to GPO modules.* |
| **Parameter 1** <input| output| inout> | *None* |
| **Parameter 2..n** |  |
| **Return Value** | Variable IDLE is returned to the light control state machine, so we can read a different can message. |
| **Precondition** | Function can be only called by the light control state machine |
| **Post condition** | *Once the light output signals have been sent to GPO, this functions sends back IDLE to the state machine and finishes the main light process.* |
| **Error Conditions** |  |



* + 1. **Function *void Main\_Ligths\_Auto\_Mode(void)***

|  |  |
| --- | --- |
| **Description** | *This function contains the logic selection of all possible Mainlights commands, day, full or off according to the signal received by the light sensor. it sends instructions to GPO modules.* |
| **Parameter 1** <input| output| inout> | *None* |
| **Parameter 2..n** |  |
| **Return Value** | None |
| **Precondition** | Function is called by the scheduler and only starts if the automatic command has been sent by the can transceiver. |
| **Post condition** | *Once the light output signals have been sent to GPO, this functions ends its process and turns the automatic flag off* |
| **Error Conditions** |  |

* + 1. **Function *void Door\_Debounce(void)***

|  |  |
| --- | --- |
| **Description** | *This function will check if the door is open, emulated with a debounced button, and will modify a variable with the door status* |
| **Parameter 1** <input| output| inout> | *None* |
| **Parameter 2..n** |  |
| **Return Value** | None |
| **Precondition** | Function is called by the scheduler and continuously checks the door status. |
| **Post condition** | *Before this functions finishes, the variable is modified to indicate the door status. It is called again by the scheduler every 3.125 ms* |
| **Error Conditions** |  |

* + 1. **Function *void Day\_Lights(T\_UBYTE lub\_On\_Off)***

|  |  |
| --- | --- |
| **Description** | *This function is in charge of controlling the GPO (lights) according the light control module. It will indicate the way the lights will turn on or off.* |
| **Parameter 1** <input| output| inout> | *Lub\_On\_Off <input>* |
| **Parameter 2..n** |  |
| **Return Value** | None |
| **Precondition** | Function is called by the lights control module, this module sends active or inactive as parameter to control the lights. |
| **Post condition** | *It turns on or off the corresponding leds according to the parameter.* |
| **Error Conditions** |  |

* + 1. **Function *void Low\_Beam\_Lights(T\_UBYTE lub\_On\_Off)***

|  |  |
| --- | --- |
| **Description** | *This function is in charge of controlling the GPO (lights) according the light control module. It will indicate the way the lights will turn on or off.* |
| **Parameter 1** <input| output| inout> | *Lub\_On\_Off <input>* |
| **Parameter 2..n** |  |
| **Return Value** | None |
| **Precondition** | Function is called by the lights control module, this module sends active or inactive as parameter to control the lights. |
| **Post condition** | *It turns on or off the corresponding leds according to the parameter.* |
| **Error Conditions** |  |

* + 1. **Function *void High\_End\_Blinker\_Lights\_Task(void)***

|  |  |
| --- | --- |
| **Description** | *This function is in charge of controlling the GPO (high-end blinker lights) according the light control module. It will indicate the way the lights will turn on or off.* |
| **Parameter 1** <input| output| inout> | *None* |
| **Parameter 2..n** |  |
| **Return Value** | None |
| **Precondition** | Function is called by the scheduler and it receives the parameter by the light control module |
| **Post condition** | *It turns on or off the corresponding leds according to the parameter values* |
| **Error Conditions** |  |

* + 1. **Function *void Luxury\_Blinker\_Lights\_Task(void)***

|  |  |
| --- | --- |
| **Description** | *This function is in charge of controlling the GPO (luxury blinker lights) according the light control module. It will indicate the way the lights will turn on or off.* |
| **Parameter 1** <input| output| inout> | *None* |
| **Parameter 2..n** |  |
| **Return Value** | None |
| **Precondition** | Function is called by the scheduler and it receives the parameter by the light control module |
| **Post condition** | *It turns on or off the corresponding leds according to the parameter values* |
| **Error Conditions** |  |

* + 1. **Function *void Standard\_Blinker\_Lights\_Task(void)***

|  |  |
| --- | --- |
| **Description** | *This function is in charge of controlling the GPO (standard blinker lights) according the light control module. It will indicate the way the lights will turn on or off.* |
| **Parameter 1** <input| output| inout> | *none* |
| **Parameter 2..n** |  |
| **Return Value** | None |
| **Precondition** | Function is called by the scheduler and it receives the parameter by the light control module |
| **Post condition** | *It turns on or off the corresponding leds according to the parameter.* |
| **Error Conditions** |  |

* + 1. **Function *void Stop\_Lights(T\_UBYTE lub\_On\_Off)***

|  |  |
| --- | --- |
| **Description** | *This function is in charge of controlling the GPO (lights) according the light control module. It will indicate the way the lights will turn on or off.* |
| **Parameter 1** <input| output| inout> | *Lub\_On\_Off <input>* |
| **Parameter 2..n** |  |
| **Return Value** | None |
| **Precondition** | Function is called by the lights control module, this module sends active or inactive as parameter to control the lights. |
| **Post condition** | *It turns on or off the corresponding leds according to the parameter.* |
| **Error Conditions** |  |

* + 1. **Function *void Standard\_Stop\_Lights(T\_UBYTE lub\_On\_Off)***

|  |  |
| --- | --- |
| **Description** | *This function is in charge of controlling the GPO (lights) according the light control module. It will indicate the way the lights will turn on or off.* |
| **Parameter 1** <input| output| inout> | *Lub\_On\_Off <input>* |
| **Parameter 2..n** |  |
| **Return Value** | None |
| **Precondition** | Function is called by the lights control module, this module sends active or inactive as parameter to control the lights. |
| **Post condition** | *It turns on or off the corresponding leds according to the parameter.* |
| **Error Conditions** |  |

* 1. **Macros**

*No complex macros are used in this software module.*

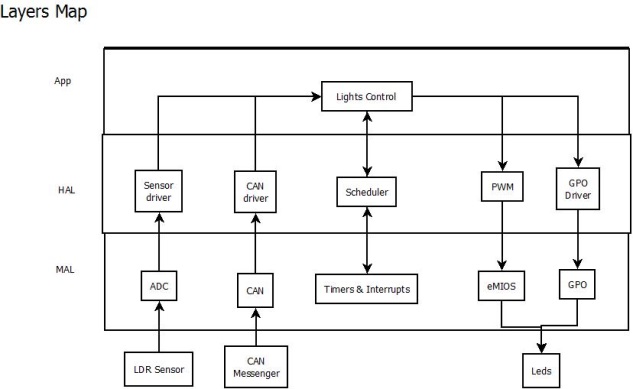
* 1. **SW module global variables**

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Memory section** | **Description** | **Shared** |
|  |
| rub\_HW\_Config | T\_UBYTE | NON\_INIT\_RAM | Hardware configuration, car version selector |  |
|  |
| rub\_Auto-Mode\_Flag | T\_UBYTE | NON\_INIT\_RAM | Automatic mode ON/OFF Flag |  |
|  |
| rub\_Messg\_ID | T\_UBYTE | NON\_INIT\_RAM | Saves the message id received by CAN |  |
|  |
| rub\_ON\_TIME | T\_UBYTE | NON\_INIT\_RAM | Saves the message parameter received by CAN |  |
|  |
| rub\_OFF\_TIME | T\_UBYTE | NON\_INIT\_RAM | Saves the message parameter id received by CAN |  |
|  |
| rub\_Main\_Lights\_Mode | T\_UBYTE | NON\_INIT\_RAM | Saves the message parameter id received by CAN |  |
|
|  |
| rub\_Open\_CktDbnc | T\_UBYTE | NON\_INIT\_RAM | Saves the door status |  |
| rub\_Parameter | T\_UBYTE | NON\_INIT\_RAM | Saves the main parameter of the CAN message received | x |
| rub\_On\_Time | T\_UBYTE | NON\_INIT\_RAM | Saves the on time parameter of the CAN message received |  |
| rub\_Off\_Time | T\_UBYTE | NON\_INIT\_RAM | Saves the off time parameter of the CAN message received |  |
| rub\_High\_End\_Flag | T\_UBYTE | NON\_INIT\_RAM | IF high end has been activated, the flag turns on |  |
| rub\_Luxury\_Flag | T\_UBYTE | NON\_INIT\_RAM | IF luxury has been activated, the flag turns on |  |
| rub\_Standard\_Flag | T\_UBYTE | NON\_INIT\_RAM | IF standard has been activated, the flag turns on |  |
| rub\_Standard\_DayL\_Flag | T\_UBYTE | NON\_INIT\_RAM | Indicates the standard day lights have been chosen |  |
| rub\_Low\_Beam\_Lights\_Flag | T\_UBYTE | NON\_INIT\_RAM | Indicates the low beam lights have been chosen |  |
| rub\_Stop\_Lights\_Flag | T\_UBYTE | NON\_INIT\_RAM | Indicates the stop lights have been chosen |  |
| rub\_Standard\_Blinker\_Status | T\_UBYTE | NON\_INIT\_RAM | Indicates the standard blinker lights status |  |
| rub\_on\_off\_Flag | T\_UBYTE | NON\_INIT\_RAM | Indicates if lights are active or inactive | x |
| rub\_On\_Timer\_flag | T\_UBYTE | NON\_INIT\_RAM | Flag to count the time the light is on | X |
| rub\_Off\_Timer\_flag | T\_UBYTE | NON\_INIT\_RAM | Flag to count the time the light is off | x |
| rub\_Turn\_Flag | T\_UBYTE | NON\_INIT\_RAM | Indicates the actual turn value (right, left or off) | X |
|  |  |  |  |  |

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

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** |
|  |  |  |