|  |  |
| --- | --- |
| **Title:** | **Window Lifter**  **SW Component < XXXXXXX >** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **History** | | | | |
| **Issue status**  (Index) | **Maturity/Date**  (draft/invalid/valid)  (dd-mmm-yyyy) | **Author**  Department | **Check/Release**  Department | **Description** |
| 1.0 | Draft  04-0ct-09 | Miguel Garcia | Miguel Garcia | Creation of the document |
| 1.1 | Draft  26-Oct-17 | Rafael Sanchez | Rafael Sánchez | Edit function Descriptions |
| 2.0 | Draft  27-Oct-17 | Rodrigo Mortera | Rodrigo Mortera | Create diagrams |

**Table of Contents**

[**1**](#_44sinio) **PURPOSE 3**

[**2**](#_2jxsxqh) **DEFINITIONS AND ABBREVIATIONS 3**

[**3**](#_z337ya) **REALIZATION CONSTRAINTS AND TARGETS 3**

[**4**](#_3j2qqm3) **SW CONCEPTUAL DESIGN 3**

[**5**](#_1y810tw) **SW COMPONENT INTERNAL BREAKDOWN 4**

[**5.1**](#_4i7ojhp) **Functional Decomposition 4**

[***5.2***](#_lnxbz9) **Function *<Type> <function name> (type par 1, .., type par n)* 5**

[***5.3***](#_2xcytpi) **Function <Type*> <function name> (type par 1, .., type par n)* 5**

# **Purpose**

*This document explains how works every component and the details of the software.*

# **Definitions and abbreviations**

**Definitions**

|  |  |
| --- | --- |
| *BlueLed* | *Define to identify the port for the Blue Led* |
| *RedLed* | *Define to identify the port for the Red Led* |
| *GreenLed* | *Define to identify the port for the Green Led* |
| *UpButton* | *Define to identify the port for the push button 12* |
| *DownButton* | *Define to identify the port for the push button 13* |
| *LedBar\_1* | *Define to identify the port for the Bar’s first Led* |
| *LedBar\_2* | *Define to identify the port for the Bar’s secondary Led* |
| *LedBar\_3* | *Define to identify the port for the Bar’s third Led* |
| *LedBar\_4* | *Define to identify the port for the Bar’s fourth Led* |
| *LedBar\_5* | *Define to identify the port for the Bar’s fifth Led* |
| *LedBar\_6* | *Define to identify the port for the Bar’s sixth Led* |
| *LedBar\_7* | *Define to identify the port for the Bar’s seventh Led* |
| *LedBar\_8* | *Define to identify the port for the Bar’s eighth Led* |
| *LedBar\_9* | *Define to identify the port for the Bar’s ninth Led* |
| *LedBar\_10* | *Define to identify the port for the Bar’s tenth Led* |
| *WDOG\_disable* |  |
| *PORT\_init* |  |
| *SOSC\_init\_8MHz* |  |
| *SPLL\_init\_160MHz* |  |
| *NormalRUNmode\_80MHz* |  |
| *LPIT0\_init* |  |
| *lpit0\_ch0\_flag\_counter;* |  |
| *lpit0\_ch1\_flag\_counter* |  |
| *clear\_GPIO* |  |
| *Push\_UpButton* |  |
| *validation\_10ms* |  |
| behavior\_UP |  |
| flag |  |
| antiPinch |  |
| Push\_DownButton |  |
| timer |  |
| behavior\_Down |  |
| Push\_DownButton |  |
| **OneTouch\_UP** |  |
| temp |  |
| switch\_flag |  |
| Manual\_up |  |
| Push\_Antipinch |  |
| clear\_Leds |  |
| **OneTouch\_Down** |  |
| Manual\_down |  |
| validation\_500ms |  |
|  |  |

**Nombre de tus funciones**

**Variables**

**Abbreviations**

|  |  |
| --- | --- |
| CH1 | Channel1 |
| CH2 | Channel2 |
| R&D | Research and Develop |

**References**

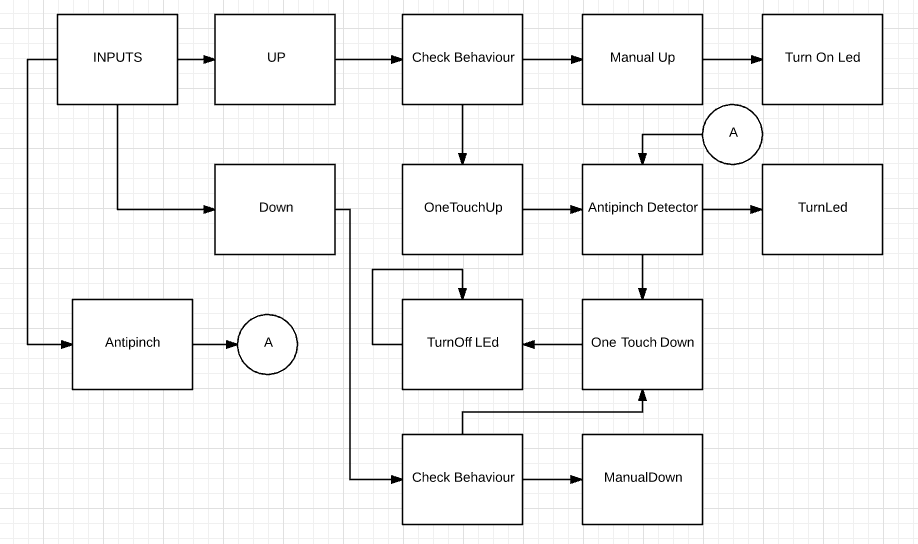
|  |  |  |
| --- | --- | --- |
| **N°** | **Document name** | **Reference** |
| *1*  *2*  *2* | *USER\_GUIDE\_S32K144*  [***S32K14x Series Reference Manual***](https://www.nxp.com/docs/en/reference-manual/S32K-RM.pdf)  [***S32K1xx Data Sheet***](https://www.nxp.com/docs/en/data-sheet/S32K-DS.pdf) | <https://www.nxp.com/products/microcontrollers-and-processors/arm-based-processors-and-mcus/s32-automotive-processors-and-microcontrollers/32-bit-automotive-general-purpose-microcontrollers:S32K?tab=Documentation_Tab> |
|  |  |  |
|  |  |  |
|  |  |  |

# **Realization constraints and targets**

Limitantes y objetivos

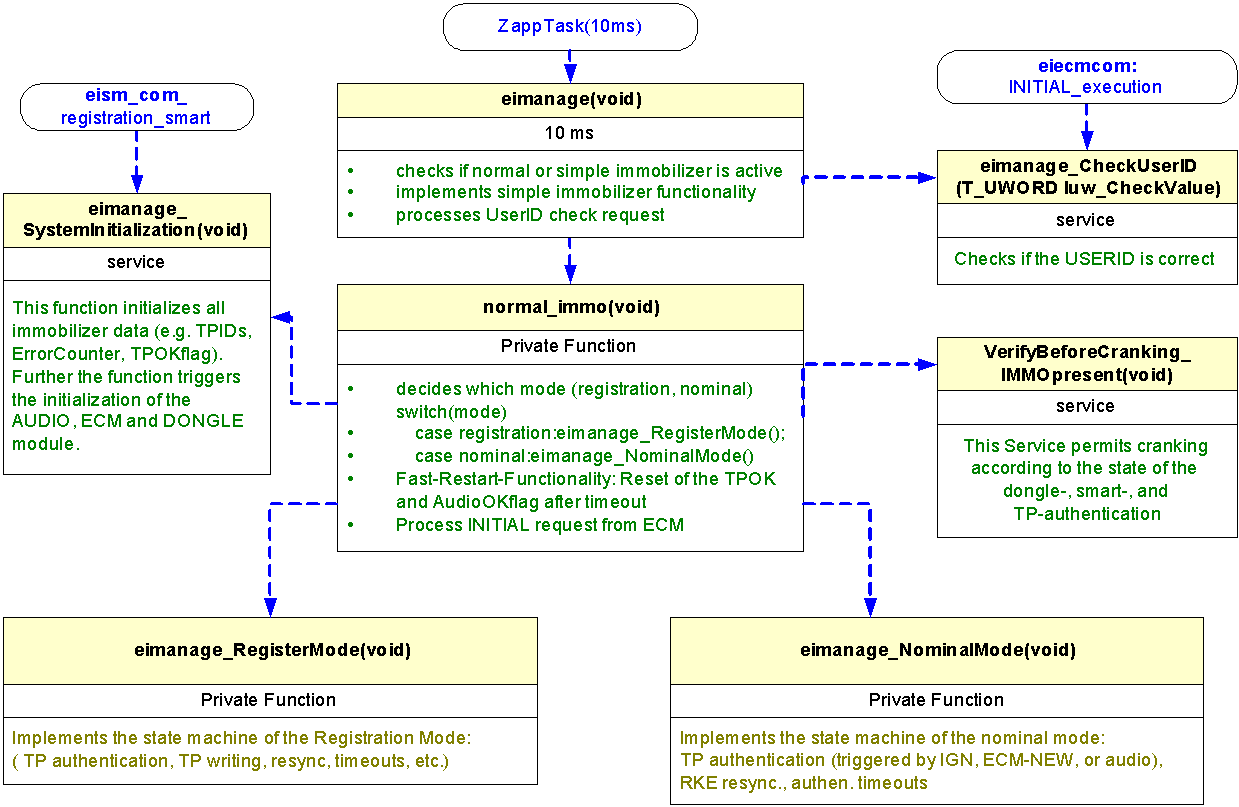
The objective of the task is to achieve to lift and down a window car simulated by 10 led bar of red color regarding to the customer request. The development of the project must be on the Development Kit Platform S32K144EVB NXP microcontroller provided by Continental R&D.

# **SW Conceptual design**



**5 SW Component internal breakdown**

## 5.1 **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*

## **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*

*Una maquina de estados o un diagrama de flujo quwe te describa el comportamiento de la funciion.*

*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.*

## **Function** v**oid Manu**al\_Up()

|  |  |
| --- | --- |
| **Description** | Ascending Turn on the Led Bar using a switch statement, the Led number is proportional with switch\_flag value. |
| **Precondition** |  |
| **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.*

## Function void Manual\_Down()

|  |  |
| --- | --- |
| **Description** | Descending Turn on the Led Bar using a switch statement, the Led number is proportional with switch\_flag value. |
| **Precondition** |  |
| **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

## Function int Push\_UpButton()

|  |  |
| --- | --- |
| **Description** | Calls Validate\_UpButton() function, to know if the Validate\_UpButton() is true or false |
| **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** | 1 , 0 |
| **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

## Function int Push\_UpDown()

|  |  |
| --- | --- |
| **Description** | Calls Validate\_DownButton() function, to know if the Validate\_DownButton() is true or false |
| **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** | 1 , 0 |
| **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

## Function int Push\_Antipinch()

|  |  |
| --- | --- |
| **Description** | Calls Validate\_Antipinch() function, to know if the Validate\_Antipinch() is true or false |
| **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** | 1 , 0 |
| **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

## Function void default\_Leds(int led)

|  |  |
| --- | --- |
| **Description** | Using int led, with a switch statement, select case 0,1,2,3 and default.  case 0 → Turn off Blue, Green and Red Led.  case 1 → Turn off Blue.  case 2 → Turn off Green.  case 3 → Turn on Red.  default → Turn on Blue, Green and Red Led. |
| **Parameter 1** <input| output| inout> | input int led |
| **Parameter 2..n** |  |
| **Return Value** | void |
| **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

## Function void clear\_GPIO()

|  |  |
| --- | --- |
| **Description** | Turn off Blue Led, Green Led, Red Led and Led Bar |
| **Parameter 1** <input| output| inout> |  |
| **Parameter 2..n** |  |
| **Return Value** | void |
| **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

## Function void timer();

|  |  |
| --- | --- |
| **Description** | When 1ms has occurred, increment lpit0\_ch1\_flag\_counter variable. |
| **Parameter 1** <input| output| inout> |  |
| **Parameter 2..n** |  |
| **Return Value** | void |
| **Precondition** | The LPIT\_MSR\_TIF1\_MASK timer should be configured and initialized |
| **Post condition** | The LPIT\_MSR\_TIF1\_MASK timer has been cleared, to know |
| **Error Conditions** |  |

**Dynamic Behavior**

State Chart1, Flow Chart1, Nassi Shneiderman