|  |  |
| --- | --- |
| **Title:** | **WINLIFT**  **SW Component < 1.0 >** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **History** | | | | |
| **Issue status**  (Index) | **Maturity/Date**  (draft/invalid/valid)  (dd-mmm-yyyy) | **Author**  Department | **Check/Release**  Department | **Description** |
| 1.0 | Draft  07-01-16 | Guillermo Ramirez | Oscar Miranda | Creation of the document |

**Table of Contents**

[1 Purpose 3](#_Toc433796033)

[2 Definitions and abbreviations 3](#_Toc433796034)

[3 Realization constraints and targets 3](#_Toc433796035)

[4 SW Conceptual design 4](#_Toc433796036)

[5 SW Component internal breakdown 5](#_Toc433796037)

[5.1 Functional Decomposition 5](#_Toc433796038)

[*5.2* *Function void Close( )* 6](#_Toc433796039)

[*5.3* *Function void Open( )* 6](#_Toc433796040)

[5.4 *Function* void blueLed\_Close( ) 7](#_Toc433796041)

[5.5 *Function* void greenLed\_Open( ) 7](#_Toc433796042)

[5.6 *Void Antipinch()* 7](#_Toc433796043)

[5.7 *Function* void countPressingTime(unsigned int counterTime ) 7](#_Toc433796044)

# Purpose

The purpose of this project is develop a software, for an embedded system, that controls a car window movement, with anti-pitch security function. The software will be implemented in a MPC5606B Freescale development board.

# Definitions and abbreviations

**Definitions**

|  |  |
| --- | --- |
| WINLIFT  GPIO  INTC  STM | Name of the project, which means Window Lifter  General purpose inputs and outputs  Interrupt Controller  System Timer Module |
|  |  |
|  |  |
|  |  |
|  |  |

**Abbreviations**

Only SW Component specific abbreviations.

**References**

|  |  |  |
| --- | --- | --- |
| **N°** | **Document name** | **Reference** |
| 1 | Traceability Matrix Template | 1 |
|  |  |  |
|  |  |  |
|  |  |  |

# Realization constraints and targets

The project has several functionalities to control the window, which includes the following ones:

* It will have a function that control the opening of the window.
* It will have a function that control the closure of the window.
* It will have an anti-pinch functionality, declared as interruption, which will stop the closure of the window and will open it. This functionality is for security purposes.
* When anti-pinch is active, a sub-function will disable all inputs for 5 seconds.
* When opening or closing the window there will be an indicator LED indicating the process in progress.
* There will be a function that counts how much time a button have been pressed.

# SW Conceptual design

*To make a conceptual design identify the product functions and the programs parts needed to produce them.*

* *Identifie all relevant parts in a system or sub-system.*
* *Define which parts are inside & outside the system boundary*
* *Define de interfaces between these parts, but also interfaces with neighboring systems and environment.*

# 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>

## 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 void Close( )***

|  |  |
| --- | --- |
| **Description** | Lifts the window |
| **Return Value** | *There is no return value* |
| **Precondition** | Only can be called when the when up button is pressed |
| **Post condition** | *Leds’ transition down-to-up executes* |
| **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.*

## ***Function void Open( )***

|  |  |
| --- | --- |
| **Description** | Lowers the window |
| **Return Value** | *There is no return value* |
| **Precondition** | Only can be called when the down button is pressed |
| **Post condition** | *Leds’ transition up-to-down executes* |
| **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 blueLed\_Close( )

|  |  |
| --- | --- |
| **Description** | Turns on the blue Led |
| **Return Value** | *There is no return value* |
| **Precondition** | Antipinch functionality is disabled |
| **Post condition** | *Turns on/off Led* |
| **Error Conditions** |  |

## ***Function***void greenLed\_Open( )

|  |  |
| --- | --- |
| **Description** | Turns on the green Led |
| **Return Value** | *There is no return value* |
| **Precondition** | Antipinch functionality is disabled |
| **Post condition** | *Turns on/off green Led* |
| **Error Conditions** |  |

## ***Void Antipinch()***

|  |  |
| --- | --- |
| **Description** | Stops and lowers the window |
| **Return Value** | *There is no return value* |
| **Precondition** | Only up button was pressed |
| **Post condition** | *Block inputs during 5 seconds* |
| **Error Conditions** |  |

## ***Function*** void countPressingTime(unsigned int counterTime )

|  |  |
| --- | --- |
| **Description** | Counts |
| **Parameter 1** <input| output| inout> | *counterTime (input)* |
| **Return Value** | *There is no return value* |
| **Precondition** | Up, down or antipinch button must have been pressed |
| **Post condition** | Call up(), down, or antipinch functions |
| **Error Conditions** |  |