# 1. Project Name:

**PO5\_LED STRING ANIMATION**

# 2. Table History:

|  |  |  |  |
| --- | --- | --- | --- |
| **Author** | **Version** | **Date** | **Change Description** |
| Walid Adel | 1.0 | 27/2/2020 | Initial Creation |
| Walid Adel | 1.1 | 28/2/2020 | Added Timer Driver Module API’s in MCAL |
| Walid Adel | 1.2 | 3/1/2020 | * Added 1- Input Output Signals   2-Software Features   * Modified Software Context Diagram * Added Req. ID and Req. Coverage to Each of API table |
| Walid Adel | 1.3 | 3/2/2020 | Modified the Software Feature Diagram and some Req. Coverage for The API’s |
| Walid Adel | 1.4 | 3/7/2020 | Covered All the Review Points that were Opened in the Review Document and also the Comments Pointed out at the last Review meeting held at 3/6/2020 |

# 3. Document Status:

|  |  |  |  |
| --- | --- | --- | --- |
| **Author** | **Version** | **Date** | **Status** |
| Walid Adel | 1.4 | 3/7/2020 | Released |

# 4. Reference Documents:

|  |  |  |  |
| --- | --- | --- | --- |
| **Reference No.** | **Document Name** | **Version** | **Status** |
| 1 | Req\_ PO5\_LED STRING ANIMATION\_CYRS.doc | 2.2 | Released |
| 2 | Req\_ PO5\_LED STRING ANIMATION\_HSI.doc | 1.5 | Released |

# 4. Table of Content:

Contents

[1. Project Name: 1](#_Toc34477932)

[2. Table History: 1](#_Toc34477933)

[3. Document Status: 1](#_Toc34477934)

[4. Reference Documents: 1](#_Toc34477935)

[4. Table of Content: 2](#_Toc34477936)

[5. Software Context Diagram: 3](#_Toc34477937)

[6. Input Output Signals: 3](#_Toc34477938)

[6.2 Input Signals 3](#_Toc34477939)

[6.2 Output Signals 3](#_Toc34477940)

[7. Software Features: 4](#_Toc34477941)

[8. Static Architecture: 4](#_Toc34477942)

[8.1 Layered Architecture Diagram: 4](#_Toc34477943)

[8.2 Layers Description: 5](#_Toc34477944)

[9. Component API’s: 6](#_Toc34477945)

[9.1 APPLICATION Layer Components: 6](#_Toc34477946)

[9.1.1 MODE MODULE 6](#_Toc34477947)

[9.1.2 TI\_LEFT MODULE 6](#_Toc34477948)

[9.1.3 TI\_RIGHT MODULE 7](#_Toc34477949)

[9.1.4 TAIL MODULE 7](#_Toc34477950)

[9.1.5 BACK LIGHTING CONTROLLER MODULE 7](#_Toc34477951)

[9.2 ECUAL Components: 8](#_Toc34477952)

[9.2.1 LED DRIVER 8](#_Toc34477953)

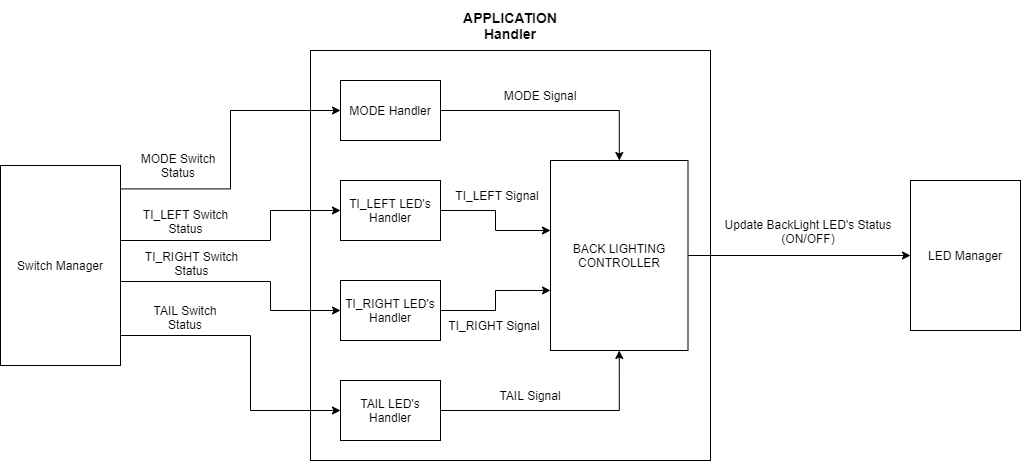
[9.2.2 SWITCH DRIVER 8](#_Toc34477954)

[9.3 MCAL Components: 8](#_Toc34477955)

[9.3.1 GPIO DRIVER 8](#_Toc34477956)

[9.3.2 TIMER DRIVER 9](#_Toc34477957)

# 5. Software Context Diagram:



# 6. Input Output Signals:

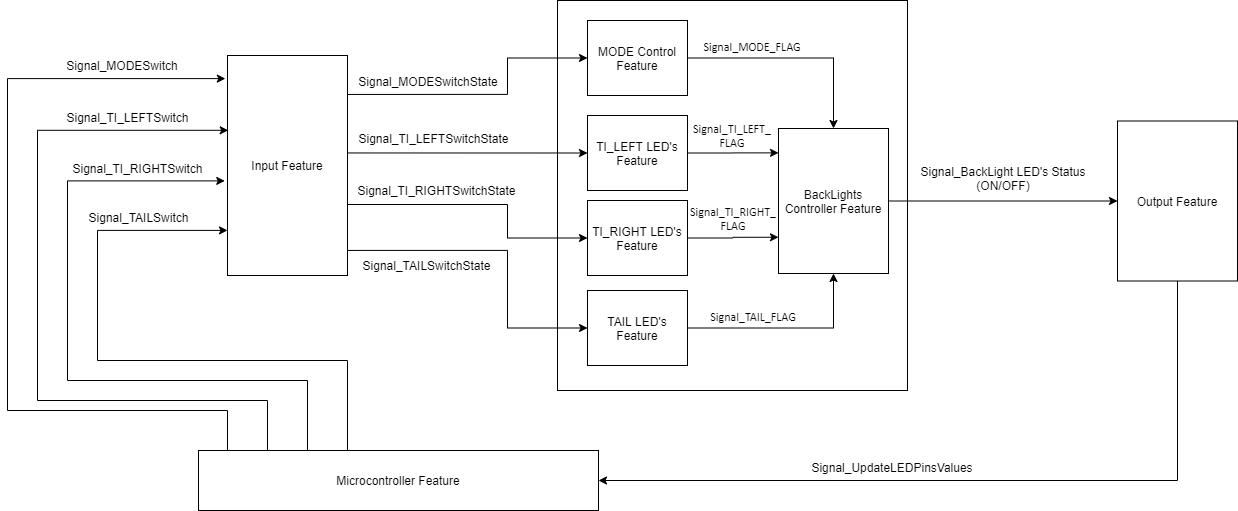
## 6.2 Input Signals

|  |  |  |  |
| --- | --- | --- | --- |
| **Signal Name** | **Signal Type** | **Unit** | **Range** |
| Signal\_MODESwitch | Input | N/A | MODE\_1 = 1, MODE\_2 = 0 |
| Signal\_TI\_LEFTSwitch | Input | N/A | ON = 1, OFF = 0 |
| Signal\_TI\_RIGHTSwitch | Input | N/A | ON = 1, OFF = 0 |
| Signal\_TAILSwitch | Input | N/A | ON = 1, OFF = 0 |

## 6.2 Output Signals

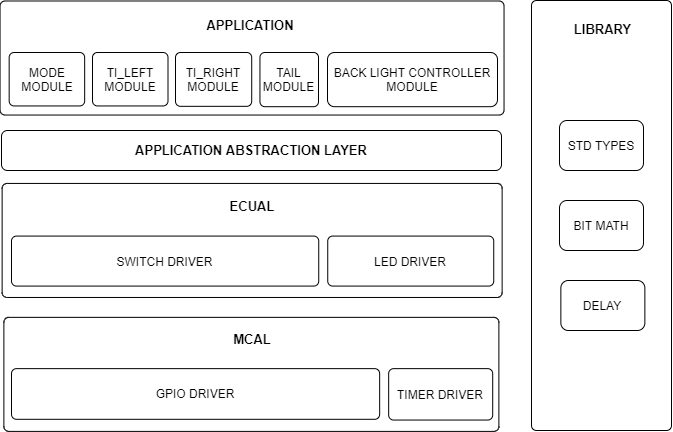
|  |  |  |  |
| --- | --- | --- | --- |
| **Signal Name** | **Signal Type** | **Unit** | **Range** |
| Signal\_UpdateLEDStatus | Output | N/A | ON = 1, OFF = 0 |

# 7. Software Features:



# 8. Static Architecture:

## 8.1 Layered Architecture Diagram:



## 8.2 Layers Description:

1-

|  |  |  |
| --- | --- | --- |
| **Layer** | **Module** | **Module Description** |
| APPLICATION | MODE MODULE | Responsible for continuously reading the MODE switch status |
| TI\_LEFT MODULE | Responsible for continuously reading the TI LEFT switch status |
| TI\_RIGHT MODULE | Responsible for continuously reading the TI RIGHT switch status |
| TAIL MODULE | Responsible for continuously reading the TAIL switch status |
| BACK LIGHTING MODULE | Responsible for analyzing all switches status and based on those inputs it will activate the TI\_LEDs and TAIL LEDs according to either MODE\_1 or MODE\_2 |

2-

|  |  |  |
| --- | --- | --- |
| **Layer** | **Module** | **Module Description** |
| APPLICATION ABSTRACTION LAYER | N/A | Responsible For Handling all The Internal Communications between All Modules in The APPLICATION Layer |

3-

|  |  |  |
| --- | --- | --- |
| **Layer** | **Module** | **Module Description** |
| ECUAL | SWITCH DRIVER | Responsible for Polling all the switches status flags in the system |
| LED DRIVER | Responsible for Updating all the LEDs state in the system |

4-

|  |  |  |
| --- | --- | --- |
| **Layer** | **Module** | **Module Description** |
| LIBRARY | STD\_TYPES | Header file that contains all variable types that will be used in the system |
| BIT\_MATH | Header file that contains all common macros that are needed in the system |
| DELAY MODULE | This Module Contains a Delay Function that is used by the timer driver to execute a delay with a specified value |

5-

|  |  |  |
| --- | --- | --- |
| **Layer** | **Module** | **Module Description** |
| MCAL | GPIO DRIVER | Responsible for:   * Configuring the pins mode. * Writing/Reading from the pins. |
| TIMER DRIVER | Responsible for :   * Configuring the timer’s modes of the system. * Starting the timer’s |

# 9. Component API’s:

## 9.1 APPLICATION Layer Components:

### 9.1.1 MODE MODULE

1-

|  |  |
| --- | --- |
| **Req. ID** | Req\_PO5\_LSAN\_GDD\_001\_V01 |
| **Req. Coverage** | Req\_PO5\_LSAN\_SRS\_002\_V02 , Req\_PO5\_LSAN\_SRS\_022\_V02 |
| **API Prototype** | Mode\_u8ERROR\_STATUSGetStatus(u8\* Switch\_Status ) |
| **Description** | This API is responsible for getting the MODE switch state |
| **Input Parameters** | u8\* Switch\_Status {Pointer to a variable at which the status of the switch will be stored, The Value Will be either PRESSED = 1 or RELEASED =0} |
| **Output Parameters** | u8 ERROR\_STATUS {OK = 1, NOK = 0} |

### 9.1.2 TI\_LEFT MODULE

1-

|  |  |
| --- | --- |
| **Req. ID** | Req\_PO5\_LSAN\_GDD\_002\_V01 |
| **Req. Coverage** | Req\_PO5\_LSAN\_SRS\_020\_V01 , Req\_PO5\_LSAN\_SRS\_021\_V01 |
| **API Prototype** | TiLeft\_u8ERROR\_STATUSGetStatus(u8\* Switch\_Status ) |
| **Description** | This API is responsible for getting the TI\_LEFT switch state |
| **Input Parameters** | u8\* Switch\_Status {Pointer to a variable at which the status of the switch will be stored, The Value Will be either PRESSED = 1 or RELEASED =0} |
| **Output Parameters** | u8 ERROR\_STATUS {OK = 1, NOK = 0} |

### 9.1.3 TI\_RIGHT MODULE

1-

|  |  |
| --- | --- |
| **Req. ID** | Req\_PO5\_LSAN\_GDD\_003\_V01 |
| **Req. Coverage** | Req\_PO5\_LSAN\_SRS\_015\_V03 , Req\_PO5\_LSAN\_SRS\_019\_V01 |
| **API Prototype** | TiRight\_u8ERROR\_STATUSGetStatus(u8\* Switch\_Status ) |
| **Description** | This API is responsible for getting the TI\_RIGHT switch state |
| **Input Parameters** | u8\* Switch\_Status {Pointer to a variable at which the status of the switch will be stored, The Value Will be either PRESSED = 1 or RELEASED =0} |
| **Output Parameters** | u8 ERROR\_STATUS {OK = 1, NOK = 0} |

### 9.1.4 TAIL MODULE

1-

|  |  |
| --- | --- |
| **Req. ID** | Req\_PO5\_LSAN\_GDD\_004\_V01 |
| **Req. Coverage** | Req\_PO5\_LSAN\_SRS\_012\_V02 , Req\_PO5\_LSAN\_SRS\_023\_V01 |
| **API Prototype** | Tail\_u8ERROR\_STATUSGetStatus(u8\* Switch\_Status ) |
| **Description** | This API is responsible for getting the TAIL switch state |
| **Input Parameters** | u8\* Switch\_Status {Pointer to a variable at which the status of the switch will be stored, The Value Will be either PRESSED = 1 or RELEASED =0} |
| **Output Parameters** | u8 ERROR\_STATUS {OK = 1, NOK = 0} |

### 9.1.5 BACK LIGHTING CONTROLLER MODULE

1-

|  |  |
| --- | --- |
| **Req. ID** | Req\_PO5\_LSAN\_GDD\_005\_V01 |
| **Req. Coverage** | Req\_PO5\_LSAN\_SRS\_003\_V02 , Req\_PO5\_LSAN\_SRS\_011\_V02 , Req\_PO5\_LSAN\_SRS\_013\_V02 , Req\_PO5\_LSAN\_SRS\_014\_V03 , Req\_PO5\_LSAN\_SRS\_016\_V03 , Req\_PO5\_LSAN\_SRS\_017\_V02 ,  Req\_PO5\_LSAN\_SRS\_018\_V01 |
| **API Prototype** | void BackLightingController(void) |
| **Description** | This API is responsible for Starting a runnable that integrate the whole Back Lighting System by:   * Analyzing the input signals from other modules to choose which mode (MODE\_1, MODE\_2) at start-up will execute * Updating the Back Lights LED’s status |
| **Input Parameters** | Void |
| **Output Parameters** | Void |

## 9.2 ECUAL Components:

### 9.2.1 LED DRIVER

1-

|  |  |
| --- | --- |
| **Req. ID** | Req\_PO5\_LSAN\_GDD\_006\_V01 |
| **Req. Coverage** | Req\_PO5\_LSAN\_SRS\_003\_V02 , Req\_PO5\_LSAN\_SRS\_011\_V02 , Req\_PO5\_LSAN\_SRS\_013\_V02 , Req\_PO5\_LSAN\_SRS\_014\_V02 , Req\_PO5\_LSAN\_SRS\_016\_V02 , Req\_PO5\_LSAN\_SRS\_017\_V02 ,  Req\_PO5\_LSAN\_SRS\_018\_V01 |
| **API Prototype** | LED\_ u8ERROR\_STATUS UpdateStatus (u8 LED\_Num , u8 status) |
| **Description** | This API is responsible for Setting the LED status to be ON/OFF |
| **Input Parameters** | 1. u8 LED\_Num {LED\_1 = 0, LED\_2 = 1, etc.} 2. u8 status {ON = 1, OFF = 0} |
| **Output Parameters** | 1. u8 ERROR\_STATUS {OK = 1, NOK = 0} |

### 9.2.2 SWITCH DRIVER

1-

|  |  |
| --- | --- |
| **Req. ID** | Req\_PO5\_LSAN\_GDD\_007\_V01 |
| **Req. Coverage** | Req\_PO5\_LSAN\_SRS\_002\_V02 , Req\_PO5\_LSAN\_SRS\_012\_V02 , Req\_PO5\_LSAN\_SRS\_015\_V02 , Req\_PO5\_LSAN\_SRS\_019\_V02 , Req\_PO5\_LSAN\_SRS\_020\_V02 , Req\_PO5\_LSAN\_SRS\_021\_V02 , Req\_PO5\_LSAN\_SRS\_023\_V02 |
| **API Prototype** | SWITCH\_ u8ERROR\_STATUSGetStatus (u8 Switch\_Num ,u8\* Status) |
| **Description** | This API is responsible Reading the status of the switch |
| **Input Parameters** | 1. u8 Switch\_Num {SWITCH\_1 = 0, SWITCh\_1 = 1, etc.} 2. u8 \*Status {Pointer to a variable at which the status of the switch will be stored, The Value Will be either PRESSED = 1 or RELEASED =0} |
| **Output Parameters** | 1. u8 ERROR\_STATUS {OK = 1, NOK = 0} |

## 9.3 MCAL Components:

### 9.3.1 GPIO DRIVER

1-

|  |  |
| --- | --- |
| **Req. ID** | Req\_PO5\_LSAN\_GDD\_008\_V01 |
| **Req. Coverage** | Req\_PO5\_LSAN\_HSI\_001\_V01 , Req\_PO5\_LSAN\_SRS\_024\_V02 |
| **API Prototype** | GPIO\_ u8ERROR\_STATUSInit(void) |
| **Description** | This API is responsible for initializing the Microcontroller GPIO Pins Direction by Writing on the DDRx Registers |
| **Input Parameters** | Void |
| **Output Parameters** | u8 ERROR\_STATUS {OK = 1, NOK = 0} |

2-

|  |  |
| --- | --- |
| **Req. ID** | Req\_PO5\_LSAN\_GDD\_009\_V01 |
| **Req. Coverage** | N/A |
| **API Prototype** | GPIO\_ u8ERROR\_STATUS WritePin(u8 Port, u8 Pin, u8 Value) |
| **Description** | This API is responsible for writing on A specified GPIO pin by Writing on the PORTx register Using the Value Entered |
| **Input Parameters** | 1. u8 Port {“A”, “B”, “C”, “D”} 2. u8 Pin {PIN\_0 = 0, PIN\_1 = 1, PIN\_2 = 2, etc.} 3. u8 Value {LOW = 0, HIGH=1} |
| **Output Parameters** | 1. u8 ERROR\_STATUS {OK = 1, NOK = 0} |

3-

|  |  |
| --- | --- |
| **Req. ID** | Req\_PO5\_LSAN\_GDD\_010\_V01 |
| **Req. Coverage** | N/A |
| **API Prototype** | GPIO\_ u8ERROR\_STATUSReadPin(u8 Port, u8 Pin, u8 \*Value) |
| **Description** | This API is responsible for writing on a specified GPIO pin.   1. Reading From the PORTx register |
| **Input Parameters** | 1. u8 Port {“A”, “B”, “C”, “D”, “E”, “F”} 2. u8 Pin {PIN\_0 = 0, PIN\_1 = 1, PIN\_2 = 2, etc.} 3. u8\* Value {Ptr to hold the Value Read From The Pin, The Value Will be either HIGH = 1 or LOW =0} |
| **Output Parameters** | 1. u8 ERROR\_STATUS {OK = 1, NOK = 0} |

### 9.3.2 TIMER DRIVER

1-

|  |  |
| --- | --- |
| **Req. ID** | Req\_PO5\_LSAN\_GDD\_011\_V01 |
| **Req. Coverage** | N/A |
| **API Prototype** | TIMER\_ u8ERROR\_STATUSInit(void) |
| **Description** | This Api is responsible for initializing a specified timer by:   1. Configuring the timer mode (PWM, Overflow, CTC) 2. Setting the pre scalar 3. Setting the preload value |
| **Input Parameters** | Void |
| **Output Parameters** | u8 ERROR\_STATUS {OK = 1, NOK = 0} |

2-

|  |  |
| --- | --- |
| **Req. ID** | Req\_PO5\_LSAN\_GDD\_012\_V01 |
| **Req. Coverage** | N/A |
| **API Prototype** | TIMER\_ u8ERROR\_STATUSStart(void) |
| **Description** | This Api is responsible for starting the timer |
| **Input Parameters** | Void |
| **Output Parameters** | u8 ERROR\_STATUS {OK = 1, NOK = 0} |