**PO2EBL\_ELECTRIC BLENDER**

**GDD DOCUMENT**

**Version 1.6**

**Proposed**

# Document Status

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Document Status** | **Author** |
| 03/09/2020 | 1.6 | Proposed | May Alaa |

# Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Change** |
| 1.0 | 02/24/2020 | Ali Samir | Initial Creation |
| 1.1 | 02/24/2020 | Mohamed M. Farag | Adding initial APIs for components |
| 1.2 | 02/29/2020 | May Alaa | Adding Requirements and APIs |
| 1.3 | 03/02/2020 | Ali Samir | Adding Software features and Software context diagram |
| 1.4 | 03/03/2020 | Mohamed M. Farag | Updating the input and output signals |
| 1.5 | 03/05/2020 | Moahmed Ibrahem | Update the static architecture |
| 1.6 | 03/09/20 | May Alaa | Updating the software context diagram Updating input/output signals  Updating requirement coverage |

Contents

[Document Status 2](#_Toc34381371)

[Document History 3](#_Toc34381372)

[Introduction 6](#_Toc34381373)

[1.1 Project Description 6](#_Toc34381374)

[1.2 Block diagram 6](#_Toc34381375)

[Software Context Diagram 7](#_Toc34381376)

[Input Output signals 8](#_Toc34381377)

[Input signals: 8](#_Toc34381378)

[Output signals: 8](#_Toc34381379)

[Software features 9](#_Toc34381380)

[Static Architecture 10](#_Toc34381381)

[**Requirements:** 12](#_Toc34381382)

[Reference table: 22](#_Toc34381383)

Table of figures:

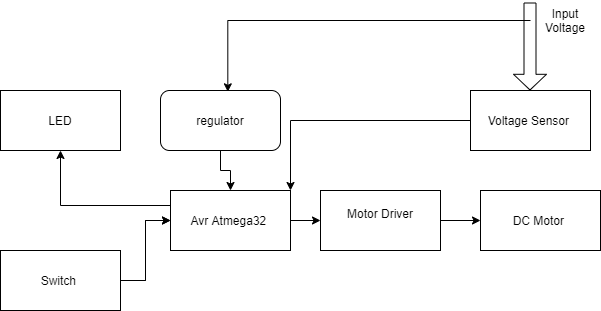
[Figure 1 Block Diagram 7](#_Toc33082078)

# Introduction

## 1.1 Project Description

The Electric Blender System is an appliance created by KENOVO. The electric blender system has 3 speeds that can be configured by the user with high safety to avoid system failure caused by unexpected voltage peaks.

## 1.2 Block diagram



\

Figure 1 Block Diagram

# Software Context Diagram

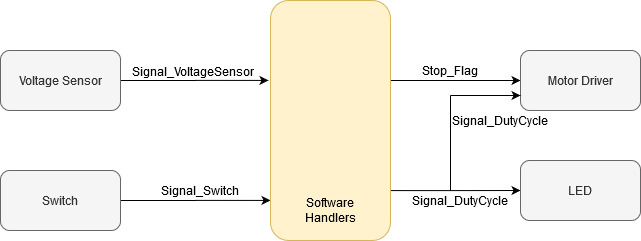


Figure Software Context Diagram

# Input Output signals

## Input signals:

|  |  |
| --- | --- |
| Signal\_ Switch | |
| Range | {0,1} |
| Unit | NA |

|  |  |
| --- | --- |
| Signal\_VoltageSensor | |
| Range | [0, 5000] |
| Unit | mV |

## Output signals:

|  |  |
| --- | --- |
| Signal\_DutyCycle | |
| Range | [0%, 100%] |
| Unit | NA |

|  |  |
| --- | --- |
| Stop\_Flag | |
| Range | {0,1} |
| Unit | NA |

# Software features

Output Feature

LED Control Features

Input Feature

Voltage\_sensor\_Status\_Signal

PWM\_Status\_Signal

Motors

Control Features

Signal switch

Voltage sensor

Microcontroller Feature

# Static Architecture

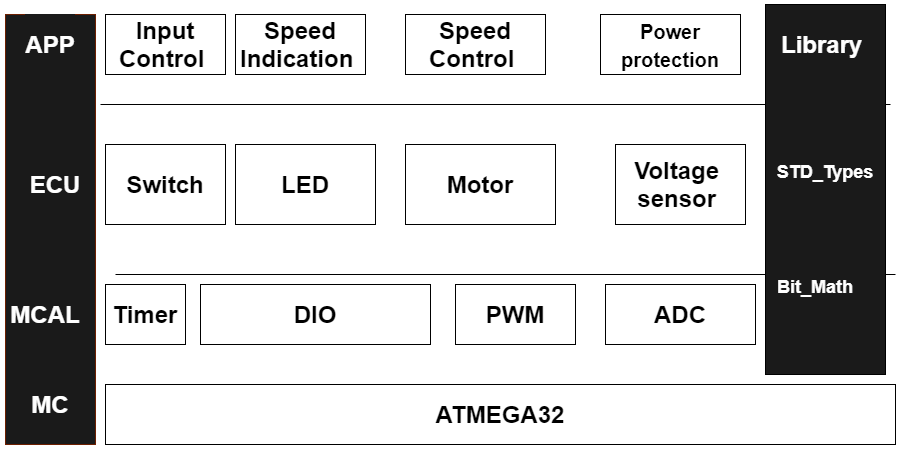


Figure Static architecture

# 

# **Requirements:**

**APPLICATION LAYER:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SPEED CONTROL** | | | |
| Requirement ID | REQ\_PO2EBL\_GDD\_1\_V01 | | |
| API | Parameters | Return values | Description |
| Error\_S Blender\_Motor\_Init(void); | void | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall Initialize the Blender’s motor. |
| Requirement ID | REQ\_PO2EBL\_GDD\_2\_V01 | | |
| Covers | REQ\_PO2EBL\_SRS\_12\_V2.0  REQ\_PO2EBL\_SRS\_13\_V2.0  REQ\_PO2EBL\_SRS\_14\_V2.0 | | |
| API | Parameters | Return values | Description |
| Error\_S Blender\_Motor\_SetSpeed(uint8\_t Speed); | uint8\_t Speed: Denotes the desired motor speed. | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall turn the blender’s motor on with desired speed. |
| Requirement ID | REQ\_PO2EBL\_GDD\_3\_V01 | | |
| Covers | REQ\_PO2EBL\_SRS\_15\_V2.0 | | |
| API | Parameters | Return values | Description |
| Error\_S Blender\_Motor \_OFF(void); | void | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall turn the Motor Off. |

|  |  |  |  |
| --- | --- | --- | --- |
| **USER INPUT CONTROLS** | | | |
| Requirement ID | REQ\_PO2EBL\_GDD\_4\_V01 | | |
| Covers | REQ\_PO2EBL\_SRS\_01\_V2.0 | | |
| API | Parameters | Return Value | Description |
| Error\_S Speed\_Switch\_Init(void); | void | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall Initialize the SPEED\_SWITCH |
| Requirement ID | REQ\_PO2EBL\_GDD\_5\_V01 | | |
| Covers | REQ\_PO2EBL\_SRS\_01\_V2.0  REQ\_PO2EBL\_SRS\_02\_V2.0  REQ\_PO2EBL\_SRS\_03\_V2.0  REQ\_PO2EBL\_SRS\_04\_V2.0  REQ\_PO2EBL\_SRS\_05\_V2.0 | | |
| API | Parameters | Return values | Description |
| Error\_S Speed\_Switch \_Read(void); | void | Error\_S: Standard Error  OK: If execution happened successfully  NOK: If Execution failed OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall read the SPEED\_SWITCH ‘s Status |

|  |  |  |  |
| --- | --- | --- | --- |
| **POWER PROTECTION** | | | |
| Requirement ID | REQ\_PO2EBL\_GDD\_6\_V01 | | |
| Covers | NA | | |
| API | Parameters | Return values | Description |
| Error\_S Blender\_Voltage\_Init(void); | void | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall Initialize the Blender’s voltage sensor |
| Requirement ID | REQ\_PO2EBL\_GDD\_7\_V01 | | |
| Covers | REQ\_PO2EBL\_SRS\_20\_V2.0  REQ\_PO2EBL\_SRS\_21\_V2.0  REQ\_PO2EBL\_SRS\_22\_V2.0 | | |
| API | Parameters | Return values | Description |
| Error\_S Blender\_Voltage \_Read(void); | void | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall get the Blender’s voltage sensor readings |

|  |  |  |  |
| --- | --- | --- | --- |
| **SPEED INDICATION** | | | |
| Requirement ID | REQ\_PO2EBL\_GDD\_8\_V01 | | |
| Covers | NA | | |
| API | Parameters | Return values | Description |
| S\_Error Speed\_LED\_Init(void); | void | S\_Error: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall Initialize the Speed indication LED. |
| Requirement ID | REQ\_PO2EBL\_GDD\_9\_V01 | | |
| Covers | REQ\_PO2EBL\_SRS\_16\_V2.0  REQ\_PO2EBL\_SRS\_17\_V2.0  REQ\_PO2EBL\_SRS\_18\_V2.0 | | |
| API | Parameters | Return values | Description |
| S\_Error Speed\_LED\_ON(void); | Void | S\_Error :Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall turn the Speed indication LED ON. |
| Requirement ID | REQ\_PO2EBL\_GDD\_10\_V01 | | |
| Covers | REQ\_PO2EBL\_SRS\_19\_V2.0 | | |
| API | Parameters | Return values | Description |
| S\_Error Speed\_LED\_OFF(void); | Void | S\_Error :Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall turn the Speed indication LED OFF. |

**ECU ABSTRACTION LAYER:**

|  |  |  |  |
| --- | --- | --- | --- |
| **LED** | | | |
| Requirement ID | REQ\_PO2EBL\_GDD\_11\_V01 | | |
| Covers | NA | | |
| API | Parameters | Return values | Description |
| Error\_S LED\_Init(u8 Lamp\_Ch\_No); | u8 LED\_Ch\_No: Denotes the Lamp to be initialized  Range: 0-255  Options:  SPEED\_INDICATION\_LED (0) | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall initialize a LED given the LED’s channel Number |
| Requirement ID | REQ\_PO2EBL\_GDD\_12\_V01 | | |
| Covers |  | | |
| API | Parameters | Return values | Description |
| Error\_S LED\_ON(u8 Lamp\_Ch\_No); | u8 LED\_Ch\_No: Denotes the Lamp to be turned on  Range: 0-255  Options:   * SPEED\_INDICATION\_LED (0) | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall turn a LED ON given the LED’s channel Number |
| Requirement ID | REQ\_PO2EBL\_GDD\_13\_V01 | | |
| Covers |  | | |
| API | Parameters | Return values | Description |
| Error\_S LED\_OFF(u8 Lamp\_Ch\_No); | u8 LED\_Ch\_No: Denotes the Lamp to be turned off  Range: 0-255  Options:  SPEED\_INDICATION\_LED (0) | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall turn a LED Off given the LED’s channel Number |

|  |  |  |  |
| --- | --- | --- | --- |
| **SWITCH** | | | |
| Requirement ID | REQ\_PO2EBL\_GDD\_14\_V01 | | |
| Covers | NA | | |
| API | Parameters | Return values | Description |
| Error\_S Switch\_Init(u8 Switch\_Ch\_No); | u8Switch\_Ch\_No: Denotes the Switch to be initialized  Range: 0-255  Options:  SPEED\_SWITCH (0) | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall Initialize a switch given the switch’s channel number |
| Requirement ID | REQ\_PO2EBL\_GDD\_15\_V01 |  |  |
| Covers | REQ\_PO2EBL\_SRS\_01\_V2.0 |  |  |
| API | Parameters | Return values | Description |
| Error\_S Switch\_Read(u8 Switch\_Ch\_No); | u8 Switch\_Ch\_No: Denotes the Switch whose status is s to be read  Range: 0-255  Options:  SPEED\_SWITCH (0) | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall read a switch’s status given the switch’s channel number |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **VOLTAGE SENSOR** |  | |  |  |
| Requirement ID | REQ\_PO2EBL\_GDD\_16\_V01 | | | |
| Covers | NA | | | |
| API | Parameters | | Return Value | Description |
| Error\_S Sensor\_Init(uint8\_t Sensor\_Ch\_No); | uint8\_t Sensor\_Ch\_No : Denotes the Sensor to be initialized  Range: 0-255  Options:  VOLTAGE\_SENSOR (0) | | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall Initialize a sensor given the sensor’s channel number |
| Requirement ID | REQ\_PO2EBL\_GDD\_17\_V01 | | | |
| Covers | REQ\_PO2EBL\_SRS\_20\_V2.0  REQ\_PO2EBL\_SRS\_21\_V2.0  REQ\_PO2EBL\_SRS\_22\_V2.0 | | | |
| API | Parameters | | Return Value | Description |
| Error\_S Sensor\_Read( uint8\_t Sensor\_Ch\_No); | uint8\_t Sensor\_Ch\_No : Denotes the Sensor to be read  Range: 0-255  Options:  VOLTAGE\_SENSOR (0) | | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall read a sensor’s reading given the sensor’s channel number |
| **MOTOR** | | | | |
| Requirement ID | | REQ\_PO2EBL\_GDD\_18\_V01 | | |
| Covers | | NA | | |
| API | | Parameters | Return Value | Description |
| Error\_S Motor\_Init(uint8\_t Motor\_Ch\_No ); | | uint8\_t Motor\_Ch\_No : Denotes the motor to be initialized  Range: 0-255  Options:  BLENDER\_MOTOR (0) | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall Initialize a MOTOR given the motor’s channel number |
| Requirement ID | | REQ\_PO2EBL\_GDD\_19\_V01 | | |
| Covers | | REQ\_PO2EBL\_SRS\_12\_V02  REQ\_PO2EBL\_SRS\_13\_V02  REQ\_PO2EBL\_SRS\_14\_V02 | | |
| API | | Parameters | Return Value | Description |
| Error\_S Motor\_SetSpeed(uint8\_t Motor\_Ch\_No, uint8\_t Motor\_Speed); | | uint8\_t Motor\_Ch\_No: Denotes the motor to be operated.  Range: 0-255  Options:  BLENDER\_MOTOR (0)  uint8\_t Motor\_Speed: Denotes the desired motor speed. | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall set the desired motor ‘s speed. |
| Requirement ID | | REQ\_PO2EBL\_GDD\_20\_V01 | | |
| API | | Parameters | Return Value | Description |
| Error\_S Motor\_Off(uint8\_t Motor\_Ch\_No); | | uint8\_t Motor\_Ch\_No: Denotes the motor to be turned off.  Range: 0-255  Options:  BLENDER\_MOTOR (0) | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall turn the motor off. |

**MICROCONTROLLER ABSTRACTION LAYER**

|  |  |  |  |
| --- | --- | --- | --- |
| **GPIO** | | | |
| Requirement ID | REQ\_PO2EBL\_GDD\_21\_V01 | | |
| Covers | NA | | |
| API | Parameters | Return values | Description |
| S\_Error GPIO\_SetPinDir(uint32\_t ui32Port, uint8\_t ui8Pins, uint32\_t ui32PinIO); | uint32\_t ui32Port: Denotes the desired port  Options:  {PORTA, PORTB, PORTC, PORTD}  uint8\_t ui8Pins: Denotes the desired pin  Options:  {PIN0..PIN7}  uint32\_t ui32PinIO: Denotes the Pin mode  Options:  {GPIO\_MODE\_IN,  GPIO\_MODE\_OUT} | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall set pins’ modes |
| Requirement ID | REQ\_PO2EBL\_GDD\_22\_V01 | | |
| Covers | NA | | |
| API | Parameters | Return values | Description |
| S\_Error GPIO\_GetPinVal(uint32\_t ui32Port, uint8\_t ui8Pins); | uint32\_t ui32Port: Denotes the desired port  Options:  {PORTA, PORTB, PORTC, PORTD}  uint8\_t ui8Pins:Denotes the desired pin  Options  {PIN0..PIN7} | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall read pins’ status |
| Requirement ID | REQ\_PO2EBL\_GDD\_23\_V01 | | |
| Covers | NA | | |
| API | Parameters | Return values | Description |
| S\_Error GPIO\_SetPinVal(uint32\_t ui32Port, uint8\_t ui8Pins, uint8\_t ui8Val); | uint32\_t ui32Port: Denotes the desired port  Options:  {PORTA, PORTB, PORTC, PORTD}  uint8\_t ui8Pins:Denotes the desired pin  Options  {PIN0..PIN7}  uint8\_t ui8Val: Denotes the desired value.  Options:  {0, 1} | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall write values to pins |

|  |  |  |  |
| --- | --- | --- | --- |
| **TIMER** | | | |
| Requirement ID | REQ\_PO2EBL\_GDD\_24\_V01 | | |
| Covers | NA | | |
| API | Parameters | Return values | Description |
| S\_Error TIMER\_Init(uint8\_t Timer\_Ch\_num); | uint8\_t Timer\_Ch\_num: Denotes the desired timer channel  The options can be:  {Timer 0, Timer 1, Timer 2} | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall initialize the timer configurations. |
| Requirement ID | REQ\_PO2EBL\_GDD\_25\_V01 | | |
| Covers | REQ\_PO2EBL\_SRS\_01\_V02  REQ\_PO2EBL\_SRS\_02\_V02  REQ\_PO2EBL\_SRS\_03\_V02 | | |
| API | Parameters | Return values | Description |
| S\_Error TIMER\_Start(uint8\_t Timer\_Ch\_num); | uint8\_t Timer\_Ch\_num: Denotes the desired timer channel  The options can be:  {Timer 0, Timer 1, Timer 2} | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall start the desired timer. |
| Requirement ID | REQ\_PO2EBL\_GDD\_26\_V01 | | |
| Covers | REQ\_PO2EBL\_SRS\_01\_V02  REQ\_PO2EBL\_SRS\_02\_V02  REQ\_PO2EBL\_SRS\_03\_V02 | | |
| API | Parameters | Return values | Description |
| S\_Error TIMER\_Stop(uint8\_t Timer\_Ch\_num); | uint8\_t Timer\_Ch\_num: Denotes the desired timer channel  The options can be:  {Timer 0, Timer 1, Timer 2} | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall stop the desired timer. |
| Requirement ID | REQ\_PO2EBL\_GDD\_27\_V01 | | |
| Covers | NA | | |
| API | Parameters | Return values | Description |
| S\_ErrorTIMER\_voidSetCallback(uint8\_t Timer\_Ch\_num , void (\*PFunc)(void)) | uint8\_t Timer\_Ch\_num: Denotes the desired timer channel  The options can be:  {Timer 0, Timer 1, Timer 2}  void (\*PFunc)(void):  void pointer to a callback function. | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall set the desired timer callback. |

|  |  |  |  |
| --- | --- | --- | --- |
| **PWM** | | | |
| Requirement ID | REQ\_PO2EBL\_GDD\_28\_V01 | | |
| Covers | NA | | |
| API | Parameters | Return Value | Description |
| S\_Error PWM\_Init(uint8\_t PWM\_Ch\_Num); | uint8\_t PWM\_Ch\_Num: Denotes the desired PWM Channel number.  The options can be:  {PWM\_0, PWM\_1} | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall initialize the desired PWM channel. |
| Requirement ID | REQ\_PO2EBL\_GDD\_29\_V01 | | |
| Covers | REQ\_PO2EBL\_SRS\_07\_V2.0  REQ\_PO2EBL\_SRS\_08\_V2.0  REQ\_PO2EBL\_SRS\_09\_V2.0  REQ\_PO2EBL\_SRS\_10\_V2.0 | | |
| API | Parameters | Return values | Description |
| S\_Error PWM\_GenerateSignal(uint8\_t PWM\_Ch\_Num, uint16\_t ONPeriod, uint16\_t TotalPeriod, uint16\_t PeakVoltage\_mV) | uint8\_t PWM\_Ch\_Num: Denotes the desired PWM Channel number.  The options can be:  {PWM\_0, PWM\_1}  uint16\_t ONPeriod: Denotes the desired ON period.  uint16\_t TotalPeriod: Denotes uint16\_t PeakVoltage\_mV | Error\_S: Standard Error  OK (=0): If execution happened successfully  NOK (=1): If Execution failed | This API shall generate a PWM signal with the desired duty cycle on the given PWM channel. |

# Reference table:

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Version** | **Document** | **Status** |
| 1 | 2.2 | SRS | Released |