**PO1\_DGC\_Digital Calculator**

**GDD Document**

**Version 1.2**

**Proposed**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Document Change History** | | | | |
| **Version** | **Author** | **Date** | **Change** | **Status** |
| 1.0 | - Alzahraa Mohamed | 24/1/2020 | * Initial creation | Draft |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Document Change History** | | | | |
| **Version** | **Author** | **Date** | **Change** | **Status** |
| 1.0 | - Alzahraa Elsallakh | 24/2/2020 | * Initial creation | Draft |
| 1.1 | - Alzahraa Elsallakh | 25/2/2020 | * Adding project Description * Adding software context diagram * Adding input and output signals * Adding software features * Adding static architecture | Proposed |
| 1.2 | - Alzahraa Elsallakh | 27/2/2020 | * Adding output signal to Read Input Control which is “Signal\_DisplayedMsg” * Changing the range of signals “Signal\_Op1” and “Signal\_Op2” from [0.0-999.99] to [0.0 - 9999999] * Changing the unit of signals “Signal\_Op1” and “Signal\_Op2” from unsigned char to float * Changing the range of “Signal\_Operator” from {A…D, \*,#} to {A,B,C,D,#} * Changing the output signal of Buzzer Control from “Signal\_RunningTone” to “ Signal\_RunningFreq” and “ Signal\_RunningDuty” * Mapping all previous changes to software feature * Adding Timer component to MCAL layer * Adding “Calculator Interface” to layered architecture * Adding all public API’s to components section | Proposed |

|  |  |  |
| --- | --- | --- |
| **Reference Documents** | | |
| **Document Name** | **Version** | **Status** |
| SRS | 1.7 | Proposed |

**Table of Contents**

[**1.** **Project Description** 8](#_Toc33707263)

[**1.1** **Software Context Diagram** 8](#_Toc33707264)

[**2.** **Input and Output Signals** 9](#_Toc33707265)

[**2.1** **Activation Control Signals** 9](#_Toc33707266)

[**2.2** **Read Input Control Signals** 10](#_Toc33707267)

[**2.3** **Buzzer Control Signals** 11](#_Toc33707268)

[**2.4** **Do Math Control Signals** 12](#_Toc33707269)

[**2.5** **Display Control Signals** 13](#_Toc33707270)

[**3.** **Software Features** 14](#_Toc33707271)

[**4.** **Static Architecture** 15](#_Toc33707272)

[**4.1** **Layered Architecture** 15](#_Toc33707273)

[**5.** **Components** 16](#_Toc33707274)

[**5.1** **MCAL API’s** 16](#_Toc33707275)

[**5.1.1** **GPIO** 16](#_Toc33707276)

[**5.1.2** **Timer** 17](#_Toc33707277)

[**5.2** **ECUAL API’s** 18](#_Toc33707278)

[**5.2.1** **KEYPAD** 18](#_Toc33707279)

[**5.2.2** **LCD** 18](#_Toc33707280)

[**5.2.3** **SWITCH** 19](#_Toc33707281)

[**5.2.4** **BUZZER** 20](#_Toc33707282)

[**5.3** **Application API’s** 21](#_Toc33707283)

[**5.3.1** **Calculator Interface** 21](#_Toc33707284)

[**5.3.2** **Activation** 21](#_Toc33707285)

[**5.3.3** **Read Input** 21](#_Toc33707286)

[**5.3.4** **Buzzer** 22](#_Toc33707287)

[**5.3.5** **Do Math** 22](#_Toc33707288)

[**5.3.6** **Display** 22](#_Toc33707289)

**Index of Figures**

[Figure ‎1‑1 Software Context Diagram 7](#_Toc33706025)

[Figure ‎2‑1 Activation Control Signals 8](#_Toc33706026)

[Figure ‎2‑2 Read Input Control Signals 9](#_Toc33706027)

[Figure ‎2‑3 Buzzer Control Signals 10](#_Toc33706028)

[Figure ‎2‑4 Do Math Control Signals 11](#_Toc33706029)

[Figure ‎2‑5 Display Control Signals 12](#_Toc33706030)

[Figure ‎3‑1 Software Features 13](#_Toc33706031)

[Figure ‎4‑1 Project Layered Architecture 14](#_Toc33706032)

# **Project Description**

This project is a digital calculator that takes input from user and displays the input and the result on a screen.

The hardware used in the calculator is Keypad which takes input from user, LCD to display the result, buzzer to generate tunes on each key press and micro controller that performs all operations in the system.

## **Software Context Diagram**

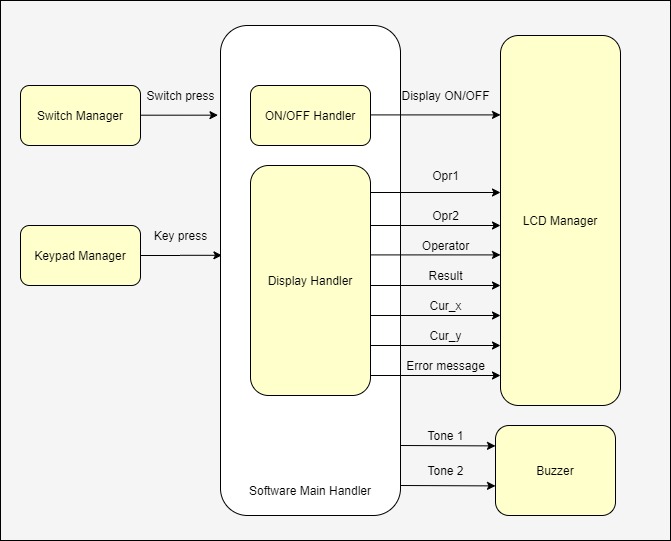


Figure ‑ Software Context Diagram

# **Input and Output Signals**

The input and output signals in the project are listed below in figures, with explained information about each signal.

## **Activation Control Signals**

Input signal: Signal\_SwitchPressed

Output signal: Signal\_OnOffState

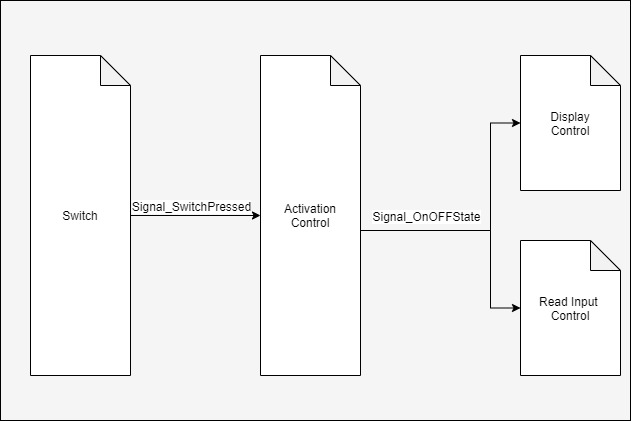


Figure ‑ Activation Control Signals

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Signal\_SwitchPressed | |  | Signal\_OnOffState | |
| Range | [0,1] |  | Range | [0,1] |
| Unit | Unsigned char |  | Unit | Unsigned char |

## **Read Input Control Signals**

Input signals: Signal\_OnOffState, Signal\_KeyPressed

Output signal: Signal\_Op1, Signal\_Op2, Signal\_Operator, Signal\_Tone

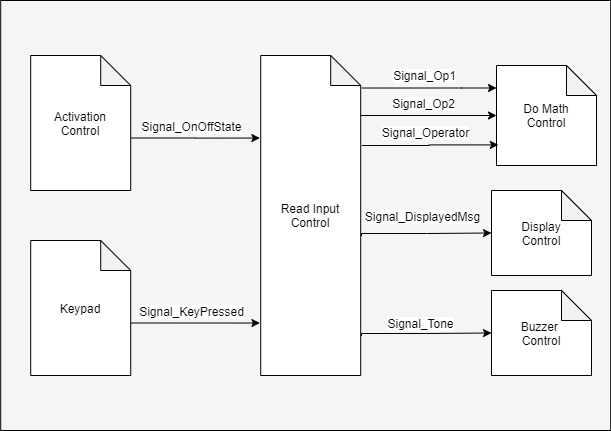


Figure ‑ Read Input Control Signals

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Signal\_OnOffState | |  | Signal\_KeyPressed | |
| Range | [0,1] |  | Range | {0…9, A…D,, #} |
| Unit | Unsigned char |  | Unit | Unsigned char |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Signal\_Op1 | |  | Signal\_Op2 | |
| Range | [0.0 – 9999999] |  | Range | [0.0 – 9999999] |
| Unit | Float |  | Unit | Float |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Signal\_Operator | |  | Signal\_Tone | |
| Range | {A,B,C,D,#} |  | Range | [1,2] |
| Unit | Unsigned char |  | Unit | Unsigned char |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Signal\_DisplayedMsg | |  |  | |
| Range | Strings, numbers, operators |  |  |  |
| Unit | Unsigned char |  |  |  |

## **Buzzer Control Signals**

Input signals: Signal\_Tone

Output signal: Signal\_RunningFreq, Signal\_RunningDuty

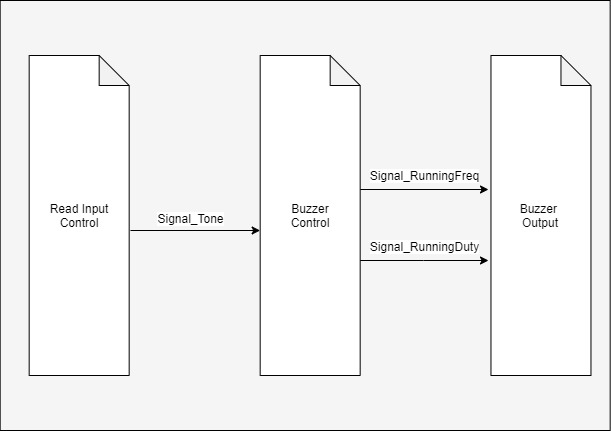


Figure ‑ Buzzer Control Signals

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Signal\_Tone | |  | Signal\_RunningFreq | |
| Range | [1,2] |  | Range | 250, 500 |
| Unit | Unsigned char |  | Unit | Unsigned short integer |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Signal\_RunningDuty | |  |  | |
| Range | 50 |  |  |  |
| Unit | Unsigned char |  |  |  |

## **Do Math Control Signals**

Input signals: Signal\_Op1, Signal\_Op2, Signal\_Operator

Output signal: Signal\_DisplayedMsg

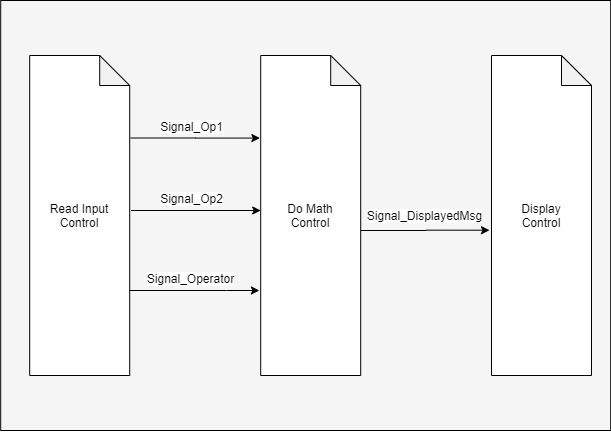


Figure ‑ Do Math Control Signals

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Signal\_Op1 | |  | Signal\_Op2 | |
| Range | [0.0 – 9999999] |  | Range | [0.0 – 9999999] |
| Unit | Float |  | Unit | Float |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Signal\_Operator | |  | Signal\_DisplayedMsg | |
| Range | {A,B,C,D,#} |  | Range | Strings, numbers, operators |
| Unit | Unsigned char |  | Unit | Unsigned char |

## **Display Control Signals**

Input signals: Signal\_OnOffState, Signal\_DisplayedMsg

Output signal: Signal\_OnOffState, Signal\_DisplayedMsg

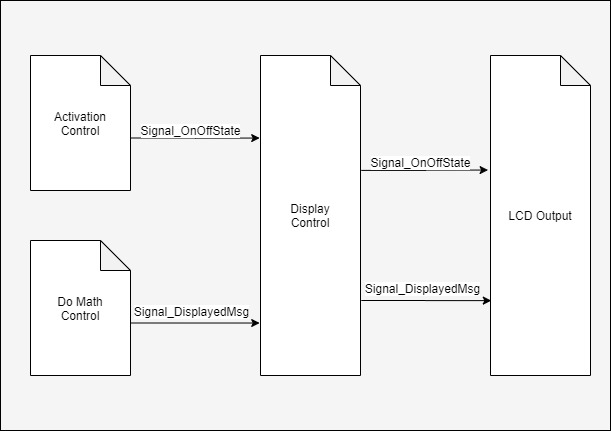


Figure ‑ Display Control Signals

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Signal\_OnOffState | |  | Signal\_DisplayedMsg | |
| Range | [0,1] |  | Range | Strings, numbers, operators |
| Unit | Unsigned char |  | Unit | Unsigned char |

# **Software Features**

The software features that maps between all software blocks and shows communication between them is shown in Figure ‎3‑1.

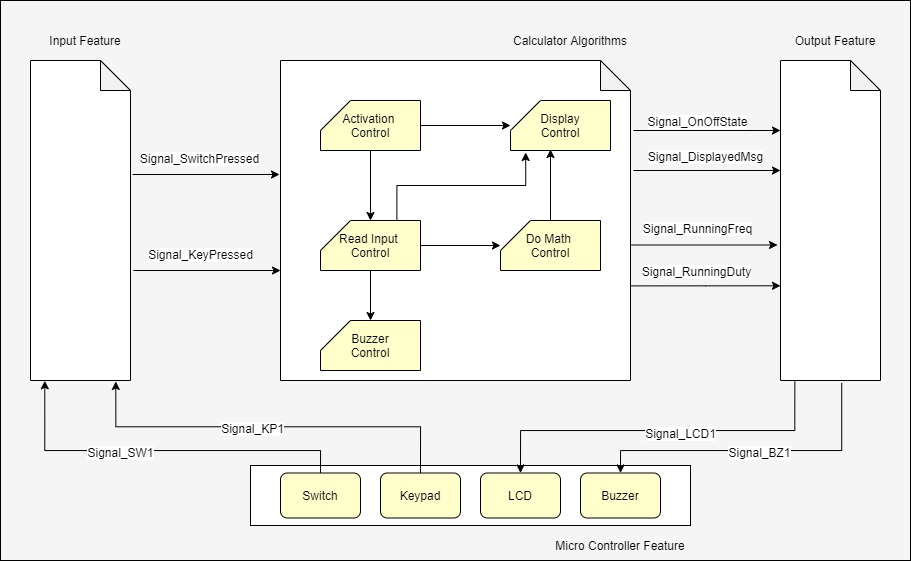


Figure ‑ Software Features

# **Static Architecture**

## **Layered Architecture**

The layered architecture represents the architecture of the project as separate horizontal layers, and shows the dependency of each module in any layer on other modules as shown in Figure ‎4‑1

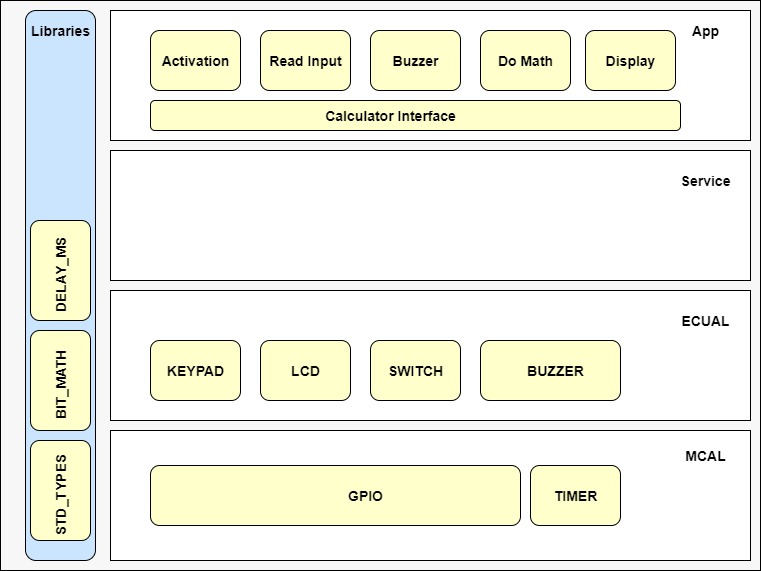


Figure ‑ Project Layered Architecture

# **Components**

Clarifying the components mentioned in the layered architecture, their public API’s, arguments, return value and requirements covered.

## **MCAL API’s**

### **GPIO**

#### GPIO Initiate

|  |  |
| --- | --- |
| API | void GPIO\_voidInit (void) |
| Description | This API shall initiate GPIO pins with initial values or configured values if exists |
| Arguments | * void |
| Return | * void |

#### GPIO Set Pin Mode

|  |  |
| --- | --- |
| API | void GPIO\_voidSetPinMode (u8 Copy\_u8Port, u8 Copy\_u8Pin, u8 Copy\_u8Mode) |
| Description | This API shall set pin mode as input or output |
| Arguments | * Copy\_u8Port: specifies pin’s port, which shall be one of this options: A, B, C, D * Copy\_u8Pin: specifies pin number, which shall be number in this range: [0-7] * Copy\_u8Mode: specifies pin mode, which is 0 for input and 1 for output |
| Return | * void |

#### GPIO Set Pin Value

|  |  |
| --- | --- |
| API | void GPIO\_voidSetPinValue (u8 Copy\_u8Port, u8 Copy\_u8Pin, u8 Copy\_u8Value) |
| Description | This API shall set pin value |
| Arguments | * Copy\_u8Port: specifies pin’s port, which shall be one of this options: A, B, C, D * Copy\_u8Pin: specifies pin number, which shall be number in this range: [0-7] * Copy\_u8Mode: specifies value set to the pin, which shall be 0 or 1 |
| Return | * void |

#### GPIO Get Pin Value

|  |  |
| --- | --- |
| API | u8 GPIO\_u8GetPinValue (u8 Copy\_u8Port, u8 Copy\_u8Pin) |
| Description | This API shall get pin value |
| Arguments | * Copy\_u8Port: specifies pin’s port, which shall be one of this options: A, B, C, D * Copy\_u8Pin: specifies pin number, which shall be number in this range: [0-7] |
| Return | * u8 that holds the value of the pin, which shall be 0 or 1 |

### **Timer**

#### Timer Initiate

|  |  |
| --- | --- |
| API | errstate TIMER\_ errstateInit (void) |
| Description | This API shall initiate timer |
| Arguments | * void |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |

#### Timer Set Callback

|  |  |
| --- | --- |
| API | errstate TIMER\_ errstateSetCallback (void(\* Copy\_PtrCallback)(void)) |
| Description | This API shall set call back function of timer interrupt |
| Arguments | * Copy\_PtrCallback: pointer that holds the address of callback function |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |

#### PWM Start

|  |  |
| --- | --- |
| API | errstate TIMER\_ errstatePWMStart (void) |
| Description | This API shall start PWM by setting callback function related to PWM |
| Arguments | * void |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |

#### PWM Generate Signal

|  |  |
| --- | --- |
| API | errstate TIMER\_ errstatePWMGenerateSignal (u16 Copy\_OnPeriod, u16 Copy\_TotalPeriod) |
| Description | This API shall generate PWM signal by setting its On period, total period and peak value |
| Arguments | * Copy\_OnPeriod: specifies the duration of the On part of the period * Copy\_TotalPeriod: specifies the duration of the total period |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |

## **ECUAL API’s**

### **KEYPAD**

#### Keypad Initiate

|  |  |
| --- | --- |
| API | errstate KEYPDAD\_errstateInit (void) |
| Description | This API shall initiate keypad on its configured pins |
| Arguments | * void |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |

#### Keypad Get Pressed Key

|  |  |
| --- | --- |
| API | errstate KEYPDAD\_errstateGetKeyPressed (u8 \* Copy\_u8PtrKey) |
| Description | This API shall get the keypad pressed key |
| Arguments | * Copy\_u8PtrKey: pointer that holds the value of the pressed key |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |

### **LCD**

#### LCD Initiate

|  |  |
| --- | --- |
| API | errstate LCD\_errstateInit (void) |
| Description | This API shall initiate LCD configurations |
| Arguments | * void |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |

#### LCD Write Data

|  |  |
| --- | --- |
| API | errstate LCD\_errstateWriteString (u8 \* Copy\_u8PtrString) |
| Description | This API shall display string on LCD |
| Arguments | * Copy\_u8PtrString: pointer that holds the string value |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |

#### LCD Write Command

|  |  |
| --- | --- |
| API | errstate LCD\_errstateWriteCmd (u8 Copy\_u8Cmd) |
| Description | This API shall send command to LCD |
| Arguments | * Copy\_ u8Cmd: specifies command to be sent to LCD |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |

#### LCD Go to Location

|  |  |
| --- | --- |
| API | errstate LCD\_errstateGoToLocation (u8 Copy\_u8Line, u8 Copy\_u8Position) |
| Description | This API shall set LCD cursor at specific line and position |
| Arguments | * Copy\_ u8Line: specifies line of the cursor, it shall be 1 or 2 * Copy\_ u8Position: specifies position of the cursor in the line, it shall be in the range [0-15] |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |

#### LCD Clear

|  |  |
| --- | --- |
| API | errstate LCD\_errstateClear (void) |
| Description | This API shall clear the display of LCD |
| Arguments | * void |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |

### **SWITCH**

#### Switch Initiate

|  |  |
| --- | --- |
| API | errstate SWITCH\_errstateInit (void) |
| Description | This API shall initiate switch on its configured pins |
| Arguments | * void |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |

#### Get Switch State

|  |  |
| --- | --- |
| API | errstate SWITCH\_errstateGetSwitchState (u8 \* Copy\_u8PtrSwitchState) |
| Description | This API shall get the switch state |
| Arguments | * Copy\_u8PtrSwitchState: pointer that holds the switch state, it shall be 1 if switch is pressed and 0 if switch is released. |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |

### **BUZZER**

#### Buzzer Initiate

|  |  |
| --- | --- |
| API | errstate BUZZER\_errstateInit (void) |
| Description | This API shall initiate buzzer on its configured pins |
| Arguments | * void |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |

#### Run Buzzer

|  |  |
| --- | --- |
| API | errstate BUZZER\_errstateRunBuzzer (u16 Copy\_u16Frequency, u8 Copy\_u8Duty) |
| Description | This API shall run buzzer for specific frequency in specific duty cycle |
| Arguments | * Copy\_u16Frequency: specifies the buzzer frequency, it shall be 250 or 500 Hz * Copy\_u8Duty: specifies the percentage of duty cycle |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |

## **Application API’s**

### **Calculator Interface**

Calculator interface is responsible of communication between application components.

### **Activation**

#### Calculator Power On

|  |  |
| --- | --- |
| API | errstate CALCULATOR\_errstatePowerOn (void) |
| Description | This API shall power the calculator on by turning LCD on and starting to accept input from keypad |
| Arguments | * void |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |
| Covers | * Req\_PO1\_DGC\_SRS\_010\_V01 |

#### Calculator Power Off

|  |  |
| --- | --- |
| API | errstate CALCULATOR\_errstatePowerOff (void) |
| Description | This API shall power the calculator off by turning LCD off and not to accept input from keypad |
| Arguments | * void |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |
| Covers | * Req\_PO1\_DGC\_SRS\_011\_V01 |

### **Read Input**

#### Keypad Input

|  |  |
| --- | --- |
| API | errstate KEYPAD\_errstateReadInput (u8 \* Copy\_u8PtrKey) |
| Description | This API shall take input from user through keypad buttons |
| Arguments | * Copy\_u8PtrKey: pointer that holds the value of the key pressed, it shall be number in the range [0-9] or letter in range [A, B, C, D]; where A represents plus operator, B represents subtract operator, C represents multiplication operator, D represents division operator, or # that represents decimal point, or \* that represents clear button. |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |
| Covers | * Req\_PO1\_DGC\_SRS\_001\_V01 |

### **Buzzer**

#### Run Tone

|  |  |
| --- | --- |
| API | errstate BUZZER\_errstateRunTone (u8 Copy\_u8Tone) |
| Description | This API shall run buzzer by specific tone |
| Arguments | * Copy\_u8Tone: specifies which tone to be run on buzzer, this shall be Tone1 or Tone2. Each tone has specific configurations for duty and frequency. |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |
| Covers | * Req\_PO1\_DGC\_SRS\_004\_V01 * Req\_PO1\_DGC\_SRS\_005\_V01 |

### **Do Math**

#### Calculate

|  |  |
| --- | --- |
| API | errstate DOMATH\_errstateCalculate (u8 Copy\_u8Op1, u8 Copy\_u8Op2, u8 Copy\_u8Operator, u8 \* Copy\_u8PtrResult) |
| Description | This API shall take operand 1, operand 2 and operation and calculate the result of the operation |
| Arguments | * Copy\_u8Op1: specifies the value of operand 1 * Copy\_u8Op2: specifies the value of operand 2 * Copy\_u8Operator: specifies the operator * Copy\_u8PtrResult: pointer that holds the value of the result |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |
| Covers | * Req\_PO1\_DGC\_SRS\_001\_V01 |

### **Display**

#### Display Message

|  |  |
| --- | --- |
| API | errstate DISPLAY\_errstateDisplayMsg (u8 Copy\_u8Mode, u8 \* Copy\_u8PtrMsg) |
| Description | This API shall display message on LCD |
| Arguments | * Copy\_u8Mode: specifies the mode of the message, it shall be 0 for error message and 1 for result message * Copy\_u8 PtrMsg: pointer that holds the string message to be displayed, it shall be error message or result message depending on mode. |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |
| Covers | * Req\_PO1\_DGC\_SRS\_007\_V01 * Req\_PO1\_DGC\_SRS\_008\_V01 |

#### Clear Display

|  |  |
| --- | --- |
| API | errstate DISPLAY\_errstateClearDisplay (void) |
| Description | This API shall clear display of LCD |
| Arguments | * void |
| Return | * errstate that holds the error state of the API implementation, it shall be OK if everything is done correctly, and NOK if anything goes wrong. |
| Covers | * Req\_PO1\_DGC\_SRS\_025\_V01 |