***Design Document.***

***Task:*** #2.

***By:*** Ahmed Taha

Contents

[1. System Description. 3](#_Toc115030808)

[2. System Architectural Pattern. 3](#_Toc115030809)

[3. System Constrains. 3](#_Toc115030810)

[4. Layered Architecture. 3](#_Toc115030811)

[5. SW Data Type Tables. 4](#_Toc115030812)

[6. SW Layers. 5](#_Toc115030813)

[I. Component Layer. 5](#_Toc115030814)

[1. Switch Module. 5](#_Toc115030815)

[2. Keypad Module. 8](#_Toc115030816)

[3. LCD Module. 11](#_Toc115030817)

# System Description.

A simple Character LCD and Keypad Interface, where the Pressed key’s Value should be Displayed on the Character LCD.

Each Press will put a key’s value in the LCD’s cursor position till the LCD is Full, then overwrite on the LCD’s data from the First location.

# System Architectural Pattern.

Monolithic Layered Architecture Pattern.

# System Constrains.

The system shall not be blocked under any condition.

# Layered Architecture.

Figure : System Layered Architecture.

App

Component

MCal

Application

Keypad

LCD

Switch

Timer

DIO

Microcontroller

* ***MCal Layer.***
* Microcontroller abstraction layer, directly control the HW peripherals inside the Controller.
* Closed Layer.
* Consist of 2 Modules:
  + Digital Input/Output.
  + Timer
* ***Component Layer.***
* The Component layer controls all the HW devices (components) on Board
* Closed Layer.
* Consist of 3 Modules:
  + LCD
  + Switch
  + Keypad.
* ***Application Layer.***
* The main app driver.
* Consist only of the main program.

# SW Data Type Tables.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | | 2 | |
| Name | uint8\_t | Name | int8\_t |
| Type | unsigned char | Type | signed char |
| Range | 0 : 255 | Range | -128 : 127 |
| Description | unsigned 8\_bit integer | Description | signed 8\_bit integer |
| 3 | | 4 | |
| Name | uint16\_t | Name | int16\_t |
| Type | unsigned short | Type | signed short |
| Range | 0 : 65535 | Range | -32768 : 32767 |
| Description | unsigned 16\_bit integer | Description | signed 16\_bit integer |
| 5 | | 6 | |
| Name | uint32\_t | Name | int32\_t |
| Type | unsigned long int | Type | signed long int |
| Range | 0 : 4294967296 | Range | -2147483648 : 2147483647 |
| Description | unsigned 32\_bit integer | Description | signed 32\_bit integer |
| 7 | | 8 | |
| Name | uint64\_t | Name | int64\_t |
| Type | unsigned long long | Type | signed long long |
| Range | 0 : (2^64) | Range | -(2^63) : (2^63) - 1 |
| Description | unsigned 64\_bit integer | Description | signed 64\_bit integer |
| 9 | | 10 | |
| Name | bool | Name | OpStatus\_t |
| Type | unsigned char | Type | unsigned char |
| Range | false = 0 true = 1 | Range | SUCCESS = 0 FAIL = 1 |
|
| Description | Boolean type for comparison | Description | the operation status |

# SW Layers.

The Detailed Design of each layer.

## Component Layer.

Component layer is a closed layer that handles and controls the on-board HW component.

### Switch Module.

#### Description.

The switch module is responsible for reading the switches status and store it into a buffer to be read with an API.

#### Data Type Tables.

|  |  |
| --- | --- |
| 1 | |
| Name | Switch\_PressState\_t |
| Type | Enumeration |
| Range | SWITCH\_NULL\_VALUE = -1 |
| KEY\_NOT\_PRESSED = 0 |
| KEY\_PRESSED = 1 |
| DEBOUNCING = 2 |
| Description | Describes the switch current status |
| 2 | |
| Name | SwitchStates\_t |
| Type | Enumeration |
| Range | FIRST\_DETECTION\_STATE = 0 |
| DEBOUNCE\_STATE = 1 |
| Description | Describes the dispatcher current state |
| 3 | |
| Name | SwitchId\_t |
| Type | unsigned char |
| Range | 0 : 255 |
| Description | describes the number of a switch |

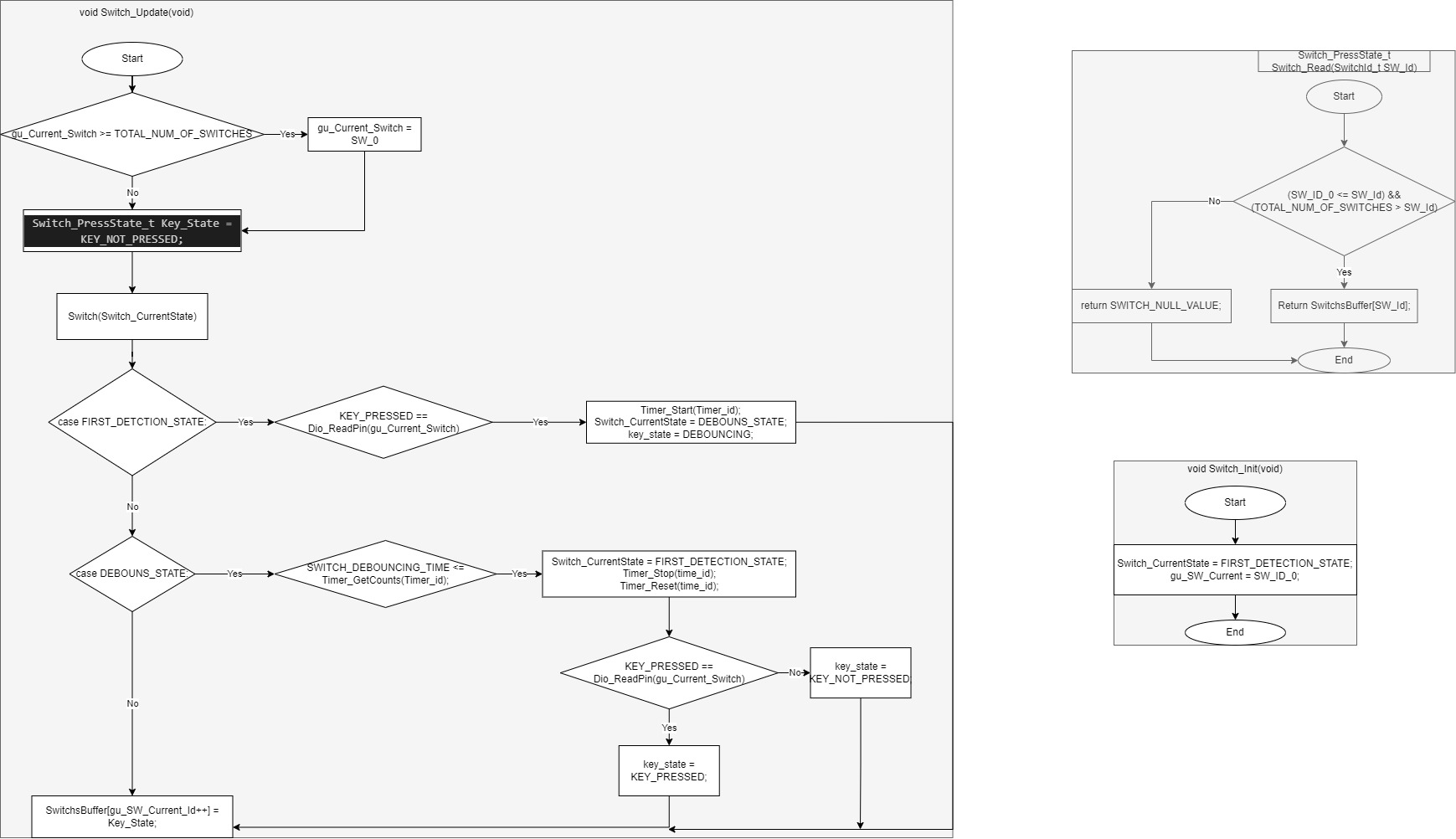
#### APIs List.

|  |  |
| --- | --- |
| Function Name | Switch\_Init |
| Arguments | None |
| Return | None |
| Synchronous | Yes |
| Reentrant | Yes |
| Description | Perform the Init sequence of the module by initialing the DIO Module with the required Switches' pins |

|  |  |
| --- | --- |
| Function Name | Switch\_Update |
| Arguments | None |
| Return | None |
| Synchronous | Yes |
| Reentrant | No |
| Description | the Module dispatcher that responsible for getting the switch press state and store it in it’s a propriate location in a buffer |

|  |  |  |  |
| --- | --- | --- | --- |
| Function Name | Switch\_Read | | |
| Arguments | Input | Name: SW\_Id | Type: SwitchId\_t |
| Description: a Switch ID to return its status | |
| Return | SWITCH\_NULL\_VALUE | | -1 |
| KEY\_NOT\_PRESSED | | 0 |
| KEY\_PRESSED | | 1 |
| DEBOUNCING | | 2 |
| Synchronous | Yes | | |
| Reentrant | No | | |
| Description | A getter API to get the status of a specific switch determined by the API argument. | | |

#### Module Flow charts.



### Keypad Module.

#### Description.

The keypad Module is responsible for detecting the pressed keys of the keypad through the switch module and return the pressed key’s data.

#### Data Type Tables.

|  |  |
| --- | --- |
| 1 | |
| Name | Keypad\_dataMatrix\_t |
| Type | uint8\_t [][] |
| Range | 0 : 255 For each cell |
|
|
|
|
|
|
|
| Description | A 2D array contains the keypad equivalent values |
| 2 | |
| Name | Keypad\_Columns\_t |
| Type | Enumeration |
| Range | From 0 : 255 according to the Columns number |
|
| Description | An Enumeration that contains the Rows Locations |
| 3 | |
| Name | Keypad\_Rows\_t |
| Type | Enumeration |
| Range | From 0 : 255 according to the Rows number2 |
| Description | An Enumeration that contains the Rows Locations |

#### APIs List.

|  |  |
| --- | --- |
| Function Name | Keypad\_Update |
| Arguments | None |
|
| Return | The Equivalent pressed Key data from [Keypad\_dataMatrix\_t] |
|
|
|
| Synchronous | Yes |
| Reentrant | No |
| Description | The Main APIs for the module which detects the pressed switch and returns its equivalent value |

#### Module Flow charts.

### LCD Module.

#### Description.

The LCD Module responsible for initializing the LCD and Display the required data on it.

#### Data Type Table.

|  |  |
| --- | --- |
| Name | LCD\_DataType\_t |
| Type | Enum |
| Range | CMD = 0 |
| DATA = 1 |
| Description | describing the write operation type |

|  |  |
| --- | --- |
| Name | LCD\_States\_t |
| Type | Enum |
| Range | LCD\_SEND\_FIRST\_NIBBLE\_STATE = 0 |
| LCD\_SEND\_SECOND\_NIBBLE\_STATE = 1 |
| LCD\_TRIGGER\_DELAY\_STATE = 2 |
| LCD\_INIT\_DELAY\_STATE = 3 |
| Description | the Module states |

#### APIs List.

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Lcd\_Write** | | |
| **Arguments** | Input | Name: Data | Type: uint8\_t |
| the data to be sent to the LCD [CMD or DATA] | |
| Input | Name: dataType | Type: LCD\_DataType\_t |
| the type of the Transmitted data is it CMD or Data | |
| **Return** | NONE | | |
| **Synchronous** | Yes | | |
| **Reentrant** | No | | |
| **Description** | A Setter API that puts a data to be sent in a buffer with its type | | |

|  |  |
| --- | --- |
| Function Name | Lcd\_Update |
| Arguments | None |
|
| Return | NONE |
|
|
| Syncronous | Yes |
| Reentrant | NO |
| Describtion | The Module Dispatcher that responsible for fetching the data to be sent to the LCD from a Buffer, and send it nibble by nibble. |

#### LCD Flow charts.