Name: Abdelrhman Walaa Hussein

LCD and KPD Design

Outline

This document shows a design for an LCD and KPD driver, using a non blocking delay, in order to initialize and run these peripherals.

The rest of the page is left as intended.

Layered Architecture

Definition

Layered Architecture (*Figure 1*) describes an architectural pattern composed of several separate horizontal layers that function together as a single unit of software.

Microcontroller Abstraction Layer (*MCAL*) is a software module that directly accesses on-chip MCU peripheral modules and external devices that are mapped to memory, and makes the upper software layer independent of the MCU.

Hardware Abstraction Layer (*HAL*) is a layer of programming that allows a computer OS to interact with a hardware device at a general or abstract level rather than at a detailed hardware level.

Design

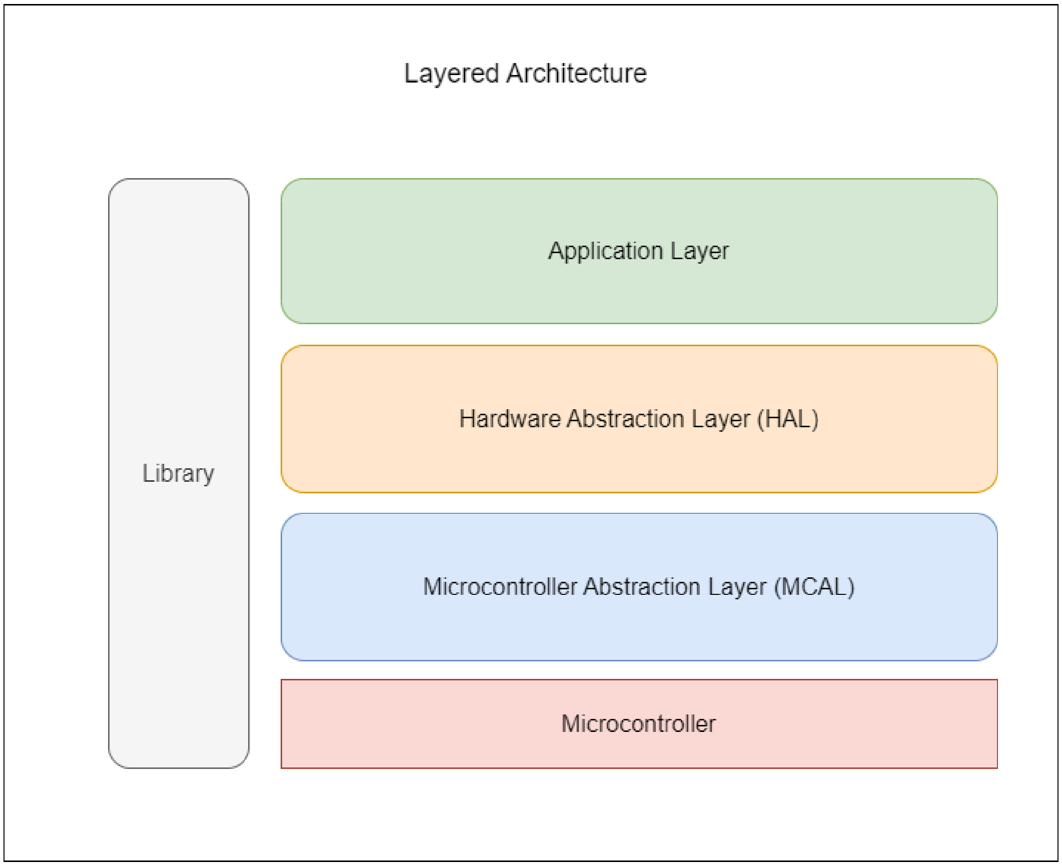

Layered Architecture
Library
Application Layer
Hardware Abstraction Layer (HAL)
Microcontroller Abstraction Layer (MCAL)
Microcontroller

Figure 1. Layered Architecture Design

Application Programming Interfaces (APIs)

Definition

An *API* is an *Application Programming Interface* that defines a set of *routines*, *protocols* and *tools* for creating an application. An *API* defines the high level interface of the behavior and capabilities of the component and its inputs and outputs.

An *API* should be created so that it is generic and implementation independent. This allows for the API to be used in multiple applications with changes only to the implementation of the API and not the general interface or behavior.

HAL APIs

A. LCD APIs

```
| Name: LCD_vdInitialization
| Input: void
| Output: void
| Description: Function to initialize ( both 4 Bit and 8 Bit Modes ) LCD
|              peripheral.
|
vd LCD_vdInitialization (void)

| Name: LCD_vdSendCmnd4BitMode
| Input: u8 Cmnd
| Output: void
| Description: Function to send a Command to LCD through Data pins.
|
vd LCD_vdSendCmnd4BitMode (u8 Cpy_u8Cmnd)

| Name: LCD_vdSendChar4BitMode
| Input: u8 Char
| Output: void
| Description: Function to send a Character to LCD through Data pins.
|
vd LCD_vdSendChar4BitMode (u8 Cpy_u8Char)

| Name: LCD_vdClearDisplay
| Input: void
| Output: void
| Description: Function to clear LCD display screen in DDRAM.
|
vd LCD_vdClearDisplay (void)
```

```
| Name: LCD_vdUpdateDelayStatus
| Input: void
| Output: void
| Description: Function to be called back when TMR reaches desired delay, it will
|              increment Delay status.
|
vd LCD_vdUpdateDelayStatus (void)
```

B. KPD APIs

```
| Name: KPD_u8GetPressedKey
| Input: Pointer to u8 ReturnedKeyValue
| Output: u8 Error or No Error
| Description: Function to check for the pressed key.
|
u8 KPD_u8GetPressedKey (u8 *Cpy_pu8ReturnedKeyValue)

| Name: KPD_vdUpdateCurrentState
| Input: void
| Output: void
| Description: Function to be called back when TMR reaches desired delay, it will
|              increment Current status.
|
vd KPD_vdUpdateCurrentState (void)
```

Flowchart Diagram

Definition

A *Flowchart* (or *Flow Chart*) is a diagram that shows the steps in a process.

Flowcharts are often used for visualizing the sequence of actions or information needed for training, documenting, planning, and decision-making. They often use symbols, shapes, and arrows to illustrate how one step leads to another.

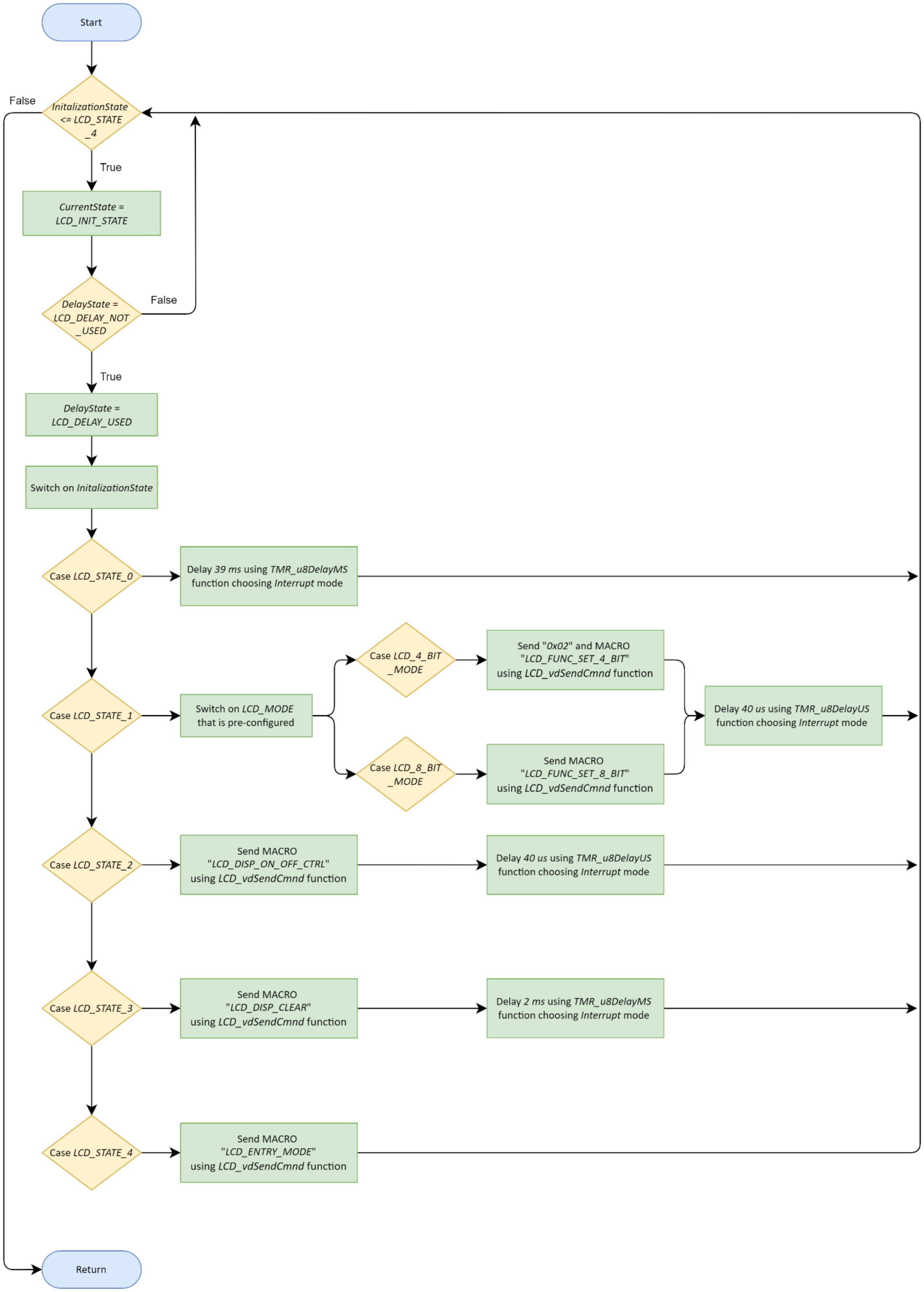
HAL Flowcharts

A. LCD Flowcharts

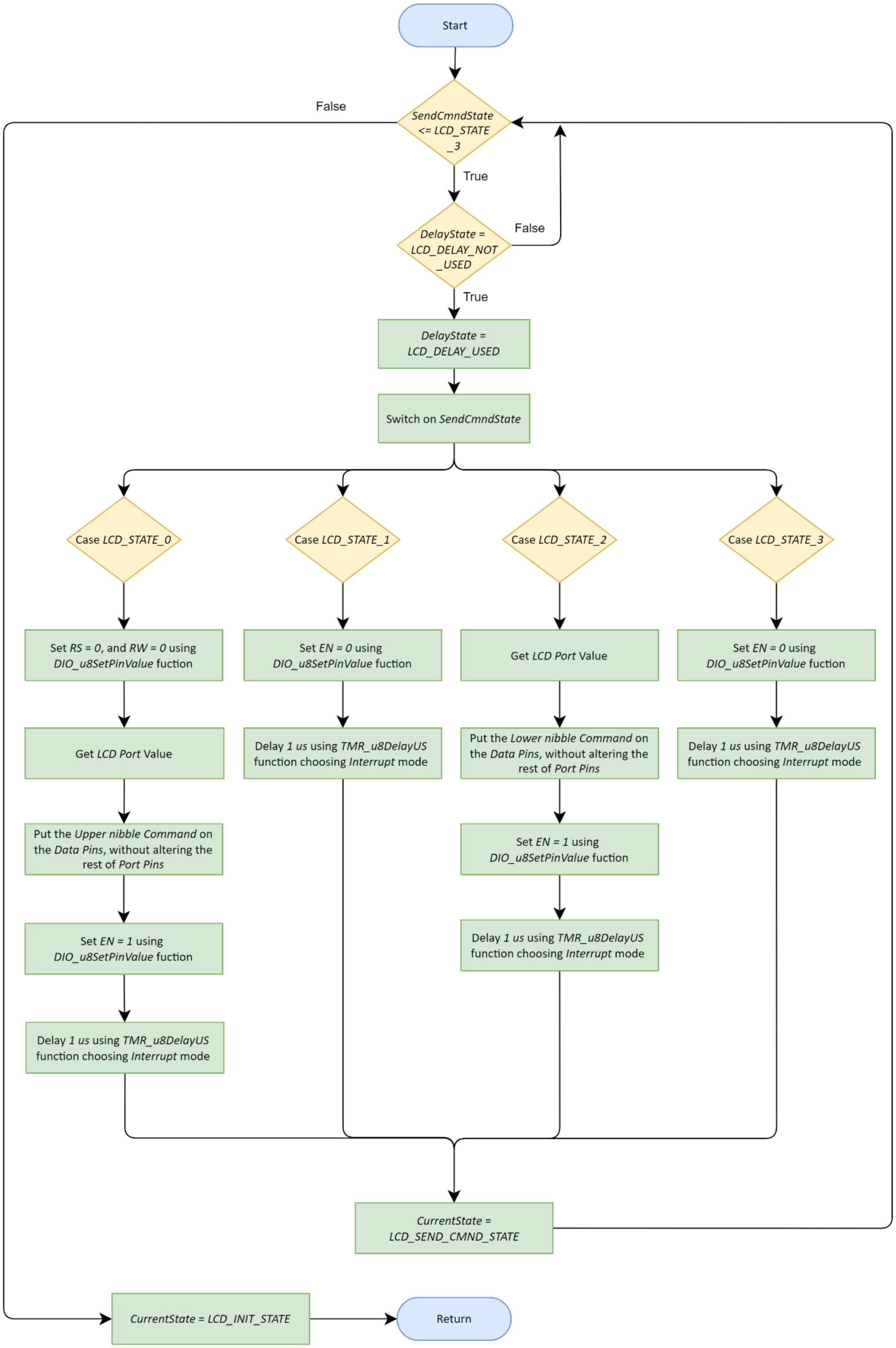
Global variables:

- DelayState = *LCD_DELAY_NOT_USED*
- InitializationState = *LCD_STATE_0*
- SendCmndState = *LCD_STATE_0*
- SendCharState = *LCD_STATE_0*
- CurrentState = *LCD_INIT_STATE*

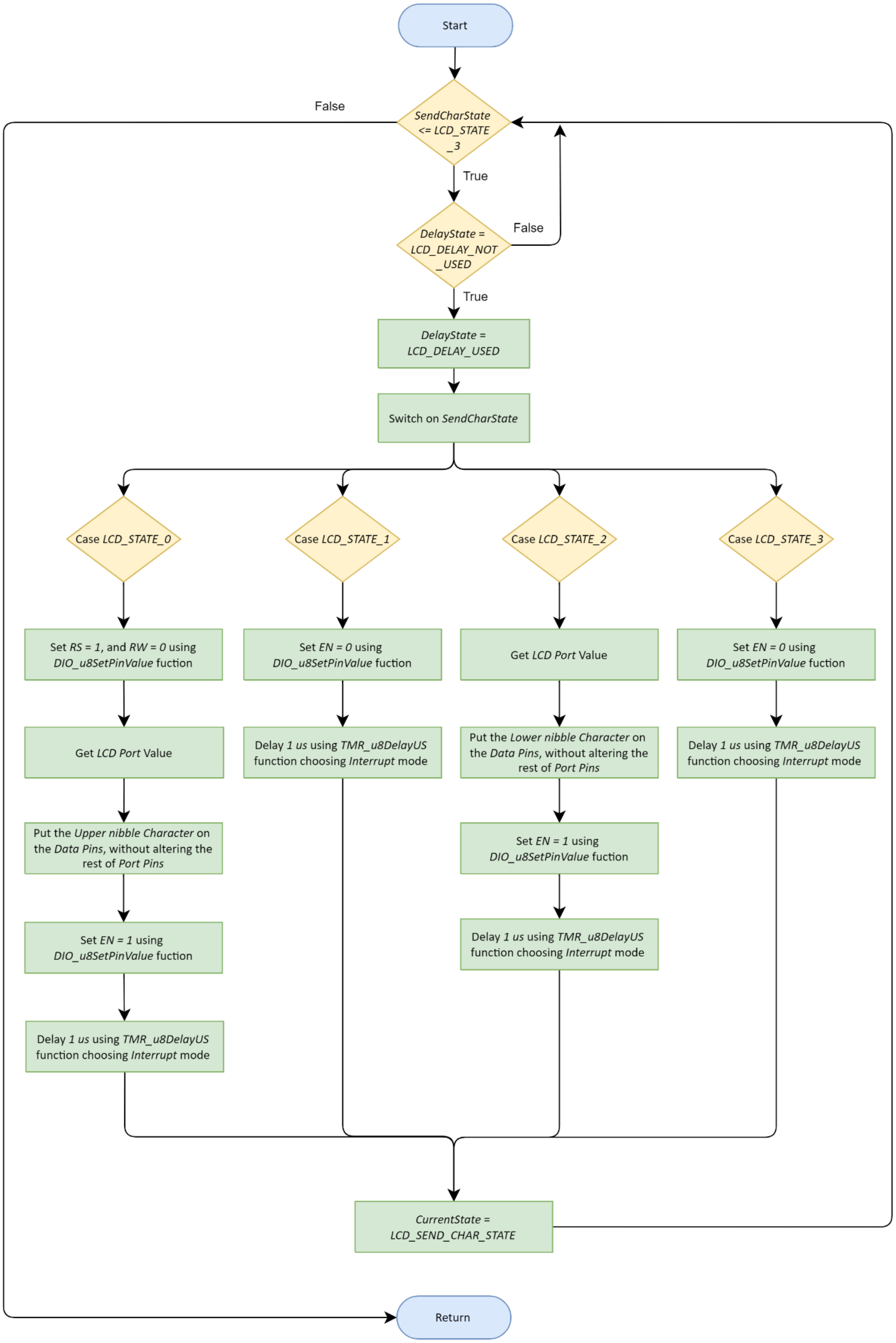
a. LCD_vdInitialization


Start
InitializationState <= LCD_STATE _4
False
True
CurrentState = LCD_INIT_STATE
DelayState = LCD_DELAY_NOT _USED
False
True
DelayState = LCD_DELAY_USED
Switch on InitalizationState
Case LCD_STATE_0
Delay 39 ms using TMR_u8DelayMS function choosing Interrupt mode
Case LCD_STATE_1
Switch on LCD_MODE that is pre-configured
Case LCD_4_BIT _MODE
Send "0x02" and MACRO "LCD_FUNC_SET_4_BIT" using LCD_vdSendCmnd function
Case LCD_8_BIT _MODE
Send MACRO "LCD_FUNC_SET_8_BIT" using LCD_vdSendCmnd function
Delay 40 us using TMR_u8DelayUS function choosing Interrupt mode
Case LCD_STATE_2
Send MACRO "LCD_DISP_ON_OFF_CTRL" using LCD_vdSendCmnd function
Delay 40 us using TMR_u8DelayUS function choosing Interrupt mode
Case LCD_STATE_3
Send MACRO "LCD_DISP_CLEAR" using LCD_vdSendCmnd function
Delay 2 ms using TMR_u8DelayMS function choosing Interrupt mode
Case LCD_STATE_4
Send MACRO "LCD_ENTRY_MODE" using LCD_vdSendCmnd function
Return

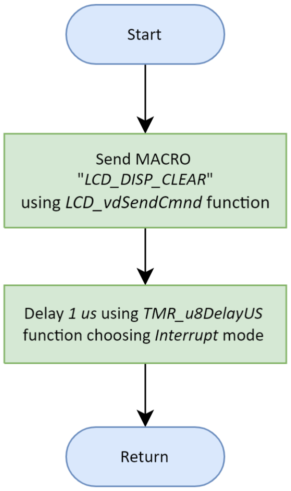
b. LCD_vdSendCmnd4BitMode


Start
SendCmndState <= LCD_STATE _3
False
True
DelayState = LCD_DELAY_NOT _USED
False
True
DelayState = LCD_DELAY_USED
Switch on SendCmndState
Case LCD_STATE_0
Case LCD_STATE_1
Case LCD_STATE_2
Case LCD_STATE_3
Set RS = 0, and RW = 0 using DIO_u8SetPinValue fuction
Get LCD Port Value
Put the Upper nibble Command on the Data Pins, without altering the rest of Port Pins
Set EN = 1 using DIO_u8SetPinValue fuction
Delay 1 us using TMR_u8DelayUS function choosing Interrupt mode
Set EN = 0 using DIO_u8SetPinValue fuction
Delay 1 us using TMR_u8DelayUS function choosing Interrupt mode
Get LCD Port Value
Put the Lower nibble Command on the Data Pins, without altering the rest of Port Pins
Set EN = 1 using DIO_u8SetPinValue fuction
Delay 1 us using TMR_u8DelayUS function choosing Interrupt mode
Set EN = 0 using DIO_u8SetPinValue fuction
Delay 1 us using TMR_u8DelayUS function choosing Interrupt mode
CurrentState = LCD_SEND_CMND_STATE
CurrentState = LCD_INIT_STATE
Return

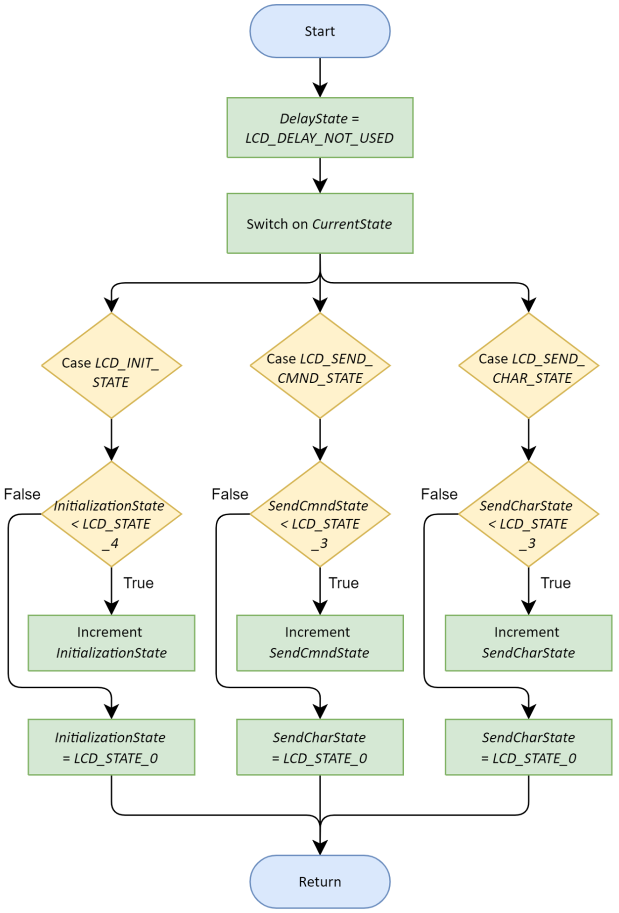
c. LCD_vdSendChar4BitMode


Start
SendCharState <= LCD_STATE _3
False
True
DelayState = LCD_DELAY_NOT _USED
False
True
DelayState = LCD_DELAY_USED
Switch on SendCharState
Case LCD_STATE_0
Case LCD_STATE_1
Case LCD_STATE_2
Case LCD_STATE_3
Set RS = 1, and RW = 0 using DIO_u8SetPinValue fuction
Set EN = 0 using DIO_u8SetPinValue fuction
Get LCD Port Value
Set EN = 0 using DIO_u8SetPinValue fuction
Get LCD Port Value
Delay 1 us using TMR_u8DelayUS function choosing Interrupt mode
Put the Lower nibble Character on the Data Pins, without altering the rest of Port Pins
Delay 1 us using TMR_u8DelayUS function choosing Interrupt mode
Put the Upper nibble Character on the Data Pins, without altering the rest of Port Pins
Set EN = 1 using DIO_u8SetPinValue fuction
Set EN = 1 using DIO_u8SetPinValue fuction
Delay 1 us using TMR_u8DelayUS function choosing Interrupt mode
Delay 1 us using TMR_u8DelayUS function choosing Interrupt mode
CurrentState = LCD_SEND_CHAR_STATE
Return

d. LCD_vdClearDisplay


Start
Send MACRO "LCD_DISP_CLEAR" using LCD_vdSendCmnd function
Delay 1 us using TMR_u8DelayUS function choosing Interrupt mode
Return

e. LCD_vdUpdateDelayStatus

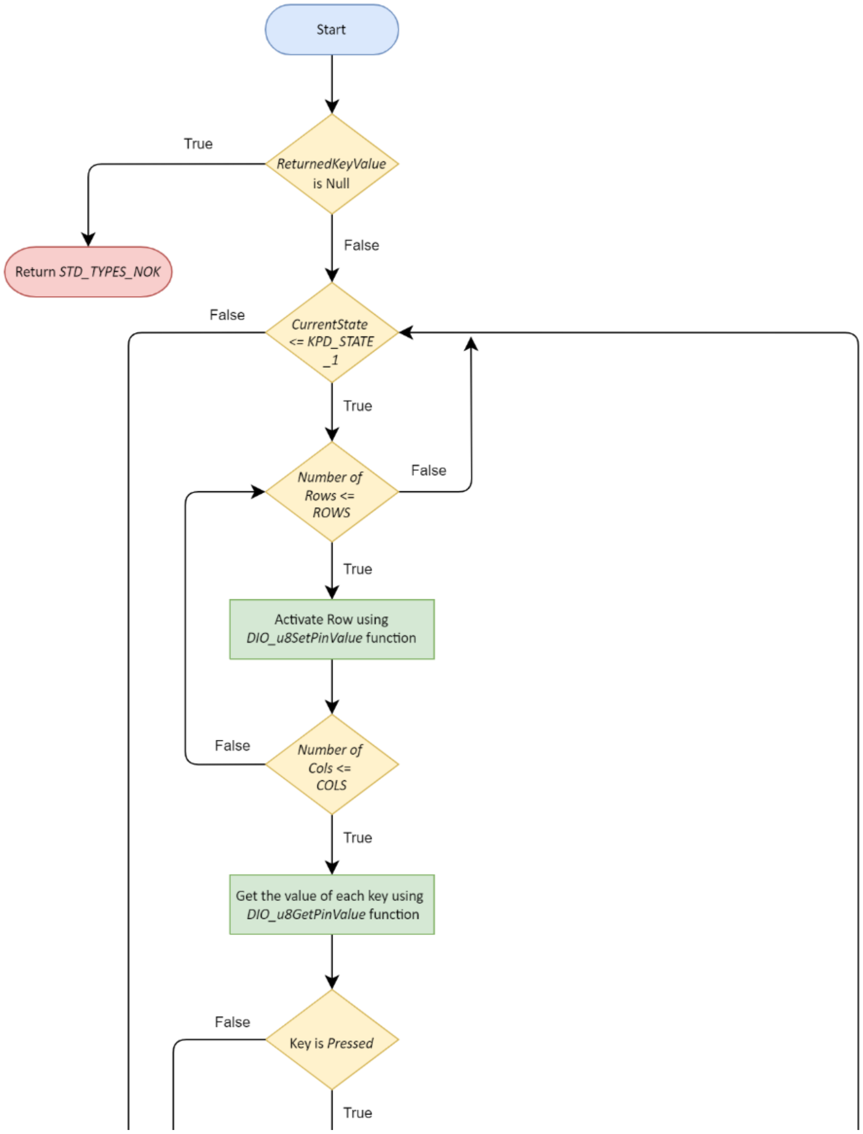

Start
DelayState = LCD_DELAY_NOT_USED
Switch on CurrentState
Case LCD_INIT_ STATE
Case LCD_SEND_ CMND_STATE
Case LCD_SEND_ CHAR_STATE
False
InitializationState < LCD_STATE _4
False
SendCmndState < LCD_STATE _3
False
SendCharState < LCD_STATE _3
True
True
True
Increment InitializationState
Increment SendCmndState
Increment SendCharState
InitializationState = LCD_STATE_0
SendCharState = LCD_STATE_0
SendCharState = LCD_STATE_0
Return

B. KPD Flowcharts

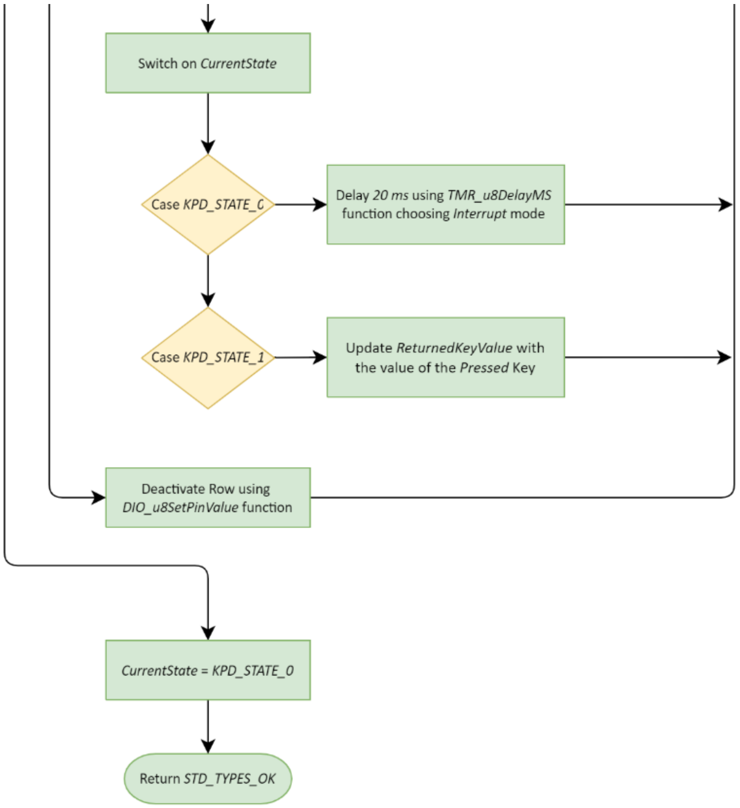
Global variables:

- CurrentState = *KPD_STATE_0*

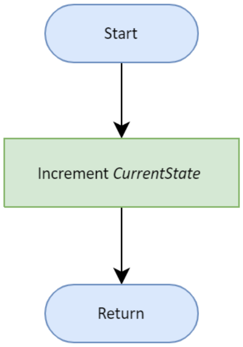
a. KPD_u8GetPressedKey



Continue the flowchart on the next page


Switch on *CurrentState*
Case *KPD_STATE_0*
Delay *20 ms* using *TMR_u8DelayMS* function choosing *Interrupt* mode
Case *KPD_STATE_1*
Update *ReturnedKeyValue* with the value of the *Pressed* Key
Deactivate Row using *DIO_u8SetPinValue* function
CurrentState = KPD_STATE_0
Return *STD_TYPES_OK*

b. KPD_vdUpdateCurrentState



References:

1. *Diagrams.net*
2. *Free Flowchart Maker - How To Create Flowcharts Online | Canva*
3. *Layered Architecture | Baeldung on Computer Science*
4. *Microcontroller Abstraction Layer (MCAL) | Renesas*
5. *Hardware Abstraction Layer - an overview | ScienceDirect Topics*
6. *What is a module in software, hardware and programming?*
7. *Embedded Basics – API's vs HAL's*