Liquid Crystal Display (LCD) Driver Design Document

1. Description:

This is a software driver for the (16 character & 2 lines)Liquid Crystal Display Actuator, this driver was developed by Anas Ebrahim at 28/3/2016 under the supervision of Eng.Mohammad Hassan and Eng.Walid El-Hennawy in the Software Engineering Course.
The driver provides the general APIs and Macros needed to use all the display options with any Microcontroller.

2. Driver Architecture:

The driver lies on the BSW Layer and contains 3 header files
1-LCD_Interface.h which contains the Functions/APIs Prototypes and variable like macros the user can use
2-LCD_Config.h which contains the configuration the user can choose to be the initial directions and values of the input output pins.
3-LCD_private.h which contains macros that is used only inside the driver.
And one source file
LCD_Prog.c which contains the Implementation of the driver APIs.
The driver also uses the DIO module in the MCAL layer and (util.h) & (types.h) library from the libraries layer.

3. Configurations:

The user should configure the 11 Microcontroller pins connected to the LCD
1-LCD_DataPort: the port (8 pins) required to write/read data to/from the LCD.
2-LCD_RSPin: the pin required to determine if the LCD is in the data (pin is high) or command (pin is low) mode.
3- LCD_RWPin: the pin required to determine if the LCD is in the Read (pin is high) or Write (pin is low) mode.
4- LCD_EnablePin: the pin required to determine if the LCD is enabled (pin is high) to write and read or disabled (pin is low).

The user also should determine if the initial state of the LCD is enabled or disabled using the macro (LCD_InitState) and the cursor direction using the macro (LCD_CursorInitDir) and if the initial state of the LCD is on or off (LCD_LCDState) and the initial state(on or off) of the cursor (LCD_Cursor) and if the cursor is blinking or not (LCD_CursorBlink)

4. APIs

1-Public:

a- LCD_voidInit():

LCD Initialization function that set the initial state of the LCD configured by user in the configuration file.

b- LCD_u8WriteCustomChar(u8 LCD_u8CopyCharArr[][8],u8 LCD_u8CopyCharCount,u8 Copy_u8X, u8 Copy_u8Y)

Write custom characters function which support the user to draw custom shapes on each character instead of the supported characters, the function takes a pointer to a 2D array which contains the binary representation of each character(LCD_u8CopyCharArr[][8]) and the number of characters in the array(LCD_u8CopyCharCount) and the position coordinates to display the characters(u8 Copy_u8X, u8 Copy_u8Y).
The function returns the state its state, u80k the array size is less than or equal to 8 and u8Error otherwise.

c- LCD_u8WriteString(u8* LCD_u8CopyString)

Write a String on the LCD function at the cursor position which takes a pointer to array of characters (LCD_u8CopyString) and returns its state, u8Error if the string size is more than 16 character and u80K otherwise.

d- LCD_u8Control(u8 Copy_u8Command)

Control the LCD appearance function which takes one of the following options:

1- LCD_CursorIncrement	2- LCD_CursorDecrement
3- LCD_Clear	4-LCD_SetCursorHome
5-LCD_TurnOn	6-LCD_TurnOff
7-LCD_TurnCursorOn	8-LCD_TurnCursorOff
9-LCD_TurnBlinkOn	10-LCD_TurnBlinkOff
11-LCD_ShiftStrRight	12-LCD_ShiftStrLeft
13-LCD_ShiftCursorRight	14-LCD_ShiftCursorLeft

The function returns its state, u8Error if the passed argument wasn't one of mentioned macros and u80K otherwise.

e- LCD_u8GoToXY(u8 Copy_u8X, u8 Copy_u8Y)

Sets the LCD cursor at the coordinate passed by the user as character number (Copy_u8X) and the line number (Copy_u8Y)
the function returns its state, u8Error if coordinate is out of boundaries (the character number is more than 16 or the line number is more than 2) and u80K otherwise.

f- LCD_voidWriteFloat(f32 Copy_f32Number)
Writes a float number (just 2 number after the floating point) function which takes a float number and writes it on the LCD.

2- Private:

a-conc(bit0,bit1,bit2,bit3,bit4,bit5,bit6,bit7)
Concatenation function like macro which takes 8 bit binary values and concatenates them into one byte.

b- LCD_voidWriteCommand(u8 LCD_u8CopyCommand)
Write Commands to the LCD to configure its appearance function which takes the command value mentioned in the datasheet.

c- LCD_voidWriteData(u8 LCD_u8CopyData)
Write data to be displayed on the LCD, function which takes the data value "in the ASCII format" to be displayed on the LCD.

5. Shared Variables

There is no shared variables in the driver

6. Integration constrains

1-The Microcontroller Pins connected to the LCD should configured as Output pins.
2-The user should clear the old data on the LCD every time before writing new data
3-The maximum number of the custom characters determined by the user is 8
4-The maximum number of characters in the string the user can write is 16 if the cursor is in the beginning.
5-The maximum coordinate is (16,1) for characters and lines respectively.

7. Hardware constrains

1-Vo Pin on the LCD should be connected with a Voltage divider resistor on a potentiometer to adjust the brightness.
2-VDD pin should be connected to 5v source.
3-VSS pin should be connected to ground.