|  |  |
| --- | --- |
|  | Prepared by:   * Merna Hany * Sara Kassem * Asmaa Adel * Mahmoud Naguib * Mohamed Ossama   Supervised by:   * Eng. Mahmoud Abo Youssef   Version:   * Global Design Document Draft: V1.2 |

**Introduction:**

This document illustrates the global design of the software used to implement the project’s functionality.

**Description:**

There are three main layers to the software, the Micro-controller layer (MCAL), the Hardware layer (HAL) and the Application layer (App.)

The document is divided into those three section with their designated functions that will be used in writing the software for the Alarm Clock project. Each section has a description of which function will use which function from another layer.

**Version:** This is version 2.1

**Modified by:** Asmaa Adel

Mahmoud Naguib

**Modifications:** Adding the description for the document, and altering the layout.

**Version Control:**

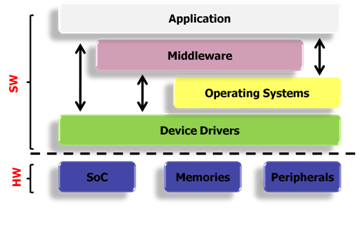
|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Modifications** |
| V 1.1 | Merna Hany  Sara Kassem  Asmaa Adel  Mahmoud Naguib  Mohamed Osama | February 12 , 2019 | First Version of the CDD, added the initial design for the software |
| V 1.2 | Merna Hany  Sara Kassem  Asmaa Adel  Mahmoud Naguib  Mohamed Osama | February 18 , 2019 | Second version of the CDD, modified some of the functions’ prototypes and the layout of the document |
| V 2.1 | Asmaa Adel  Mahmoud Naguib | March 13 , 2019 | Third version of the CDD, added the static architecture layers, and modified the layout of the tables of the API’s |

**Static Design:**

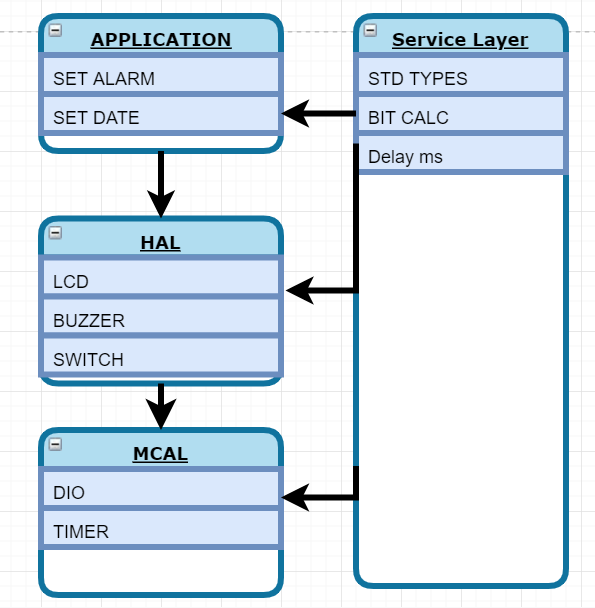
**Description:**

The layered architecture is a way to design the software in a modular way. There are main three layers, each layer can use API’s from the previous layer but not vice versa.

The following chart shows the architecture layers as a general overview.



The following chart explains the architecture used and the calling pattern.



1. Micro-controller Layer (MCAL):

Which has two main API’s that directly deals with the hardware of the micro-controller. The description of these API’s is as follows.

1. Digital Input/Output
2. DIO\_voidIntialize
3. DIO\_u8SetPinDirection
4. DIO\_u8SetPinValue

|  |  |
| --- | --- |
| 1. **ID** | **CDD\_001** |
| **Prototype** | void DIO\_voidIntialize(void) |
| **Return** | None |
| **Arguments** | None |
| **Description** | This function is aim to make the initial configuration as per the project need. |

|  |  |
| --- | --- |
| **ID** | **CDD\_002** |
| **Prototype** | u8 DIO\_u8SetPinDirection(u8 Copy\_u8PinId, u8 Copy\_u8PinDirection) |
| **Return** | Local\_u8Errorstate: The result of the input validation state |
| **Arguments** | Copy\_u8PinId: determine the pin number which is involved in the operation.  Copy\_u8PinDirection: determine the pin direction to apply a certain task to this pin in HAL component handlers. |
| **Description** | This function is aim to make change the I/O pin direction so that it can perform a certain functionality. |

|  |  |
| --- | --- |
| **ID** | **CDD\_003** |
| **Prototype** | u8 DIO\_u8SetPinValue(u8 Copy\_u8PinId, u8 Copy\_u8PinValue) |
| **Return** | Local\_u8Errorstate: The result of the input validation state |
| **Arguments** | Copy\_u8PinId: determine the pin number which is involved in the operation.  Copy\_u8PinValue: determine the pin value to apply in the pin output register. |
| **Description** | This function is aim to make change the I/O pin value to carry out a certain user interface through one of the HAL component. |

|  |  |
| --- | --- |
| **ID** | **CDD\_004** |
| **Prototype** | u8 DIO\_u8GetPinValue(u8 Copy\_u8PinId, u8\* Copy\_Pu8PinValue) |
| **Return** | Local\_u8Errorstate: The result of the input validation state |
| **Arguments** | Copy\_u8PinId: determine the pin number which is involved in the operation.  Copy\_Pu8PinValue: Pointer to a user defined address such that it save the pin current input value in it. |
| **Description** | This function is aim to get the user input and pass it to the system to behave accordingly. |

1. Timer
2. Timer0
3. ISR

|  |  |
| --- | --- |
| 1. **ID** | **CDD\_010** |
| **Prototype** | void timer0(u8 Copy\_u8prescaler, u8 Copy\_u8ticks, void (\*Copy\_TimerFunc)(void)) |
| **Return** | None |
| **Arguments** | Copy\_u8prescaler: used for making a prescale for the clock cycle frequency source.  Copy\_u8ticks: used to define the desired maximum value for the counter which it cannot overflow.  Copy\_TimerFunc: this is the function used at the timer interrupt handler and the function is used as a call back function setter. |
| **Description** | This function to initialize the timer, set the maximum value which the timer cannot exceed and the function to implement when responding the handler. |

|  |  |
| --- | --- |
| **ID** | **CDD\_011** |
| **Prototype** | ISR(TIMER0\_COMPA\_vect) |
| **Return** | None |
| **Arguments** | None |
| **Description** | This function to be called by hardware to perform the task which the user had assign to at the previous function. |

1. Hardware Layer (HAL):
2. LCD:
3. LCD\_Initialize
4. Write\_character
5. Write\_Command

|  |  |
| --- | --- |
| **ID** | **CDD\_101** |
| **Prototype** | void LCD\_Initailize(void) |
| **Return** | None |
| **Arguments** | None |
| **Description** | This function is aim to make the initial configuration for the LCD and to consider the LCD hardware initialization sequence which is mentioned in datasheet. |

|  |  |
| --- | --- |
| **ID** | **CDD\_102** |
| **Prototype** | void LCD\_voidWriteCommand(u8 Copy\_u8command) |
| **Return** | None |
| **Arguments** | Copy\_u8command: argument carrying a predefined hex value which has a certain interpretation according to the data sheet. |
| **Description** | This function is used to send LCD command setting the RS and RW pin to the suitable value for this task. |

|  |  |
| --- | --- |
| **ID** | **CDD\_103** |
| **Prototype** | void LCD\_voidWriteCharatacter(u8 Copy\_u8Data) |
| **Return** | None |
| **Arguments** | u8 Copy\_u8Data: argument carrying the ASCII code corresponding to the character that will appear to the user. |
| **Description** | This function is used to send ASCII code that will display a character into the screen. |

|  |  |
| --- | --- |
| **ID** | **CDD\_104** |
| **Prototype** | u8 LCD\_voidWriteData(u8 \*Copy\_Pu8Data,u8 Copy\_XPos,u8 Copy\_YPos) |
| **Return** | Local\_u8ErrorState: value reflect the input argument validation |
| **Arguments** | Copy\_Pu8Data: Pointer to array of character ended with null terminator.  Copy\_XPos: This argument for the horizontal position on the screen.  Copy\_YPos: This argument for the vertical position on the screen. |
| **Description** | This function is aim to make the initial configuration as per the project need. |

1. Push Button
2. Is\_pressed

|  |  |
| --- | --- |
| **ID** | **CDD\_111** |
| **Prototype** | u8 SWITCH\_u8GetSwitchState(u8 Copy\_u8SwitchId,u8\* Copy\_Pu8SwitchState); |
| **Return** | Local\_u8ErrorState: Validate the user input value for switch id |
| **Arguments** | Copy\_u8SwitchId : This is argument represent the ID of the specific switch from a multiple available switch to be configured. |
| **Description** | This function is aim to get the switch state if it is pressed or not. |

1. Buzzer:
2. Buzzer\_voidStart
3. Buzzer\_voidStop

|  |  |
| --- | --- |
| 1. **ID** | **CDD\_121** |
| **Prototype** | void Buzzer\_voidStart(void) |
| **Return** | None |
| **Arguments** | None |
| **Description** | This function is aim to latch the buzzer to generate voice. |

|  |  |
| --- | --- |
| **ID** | **GDD\_122 >>>> CDD\_122** |
| **Prototype** | void Buzzer\_voidStop(void) |
| **Return** | None |
| **Arguments** | None |
| **Description** | This function is aim to make the buzzer stop. |

Application (App):

1. Set\_alarm

|  |  |
| --- | --- |
| **Function Name** | * Start |
| **Function Prototype** | U8 ALARM\_u8SetAlarm (u8 Copy\_u8PinNumber) |
| **Inputs** | * The value of the alarm to be set |
| **Outputs** | * Error State |
| **Description** | The function is supposed to set the alarm with a given value |
| **Calls** | * DIO\_SetPinValue * LCD\_u8WriteCharacter |
| **Coverage** | * CDD\_121 |

1. Set\_date

|  |  |
| --- | --- |
| **Function Prototype** | U8 ALARM\_u8SetAlarm (u8 Copy\_u8PinNumber) |
| **Inputs** | * The value of the date to be set |
| **Outputs** | * Error State |
| **Description** | The function is supposed to set the date with a given value |
| **Function Prototype** | U8 ALARM\_u8SetDate (u8 Copy\_u8PinNumber) |
| **Calls** | * DIO\_SetPinValue * LCD\_u8WriteCharacter |
| **Coverage** | * CDD\_122 |

**Initial configuration:**

The following table denoted the pin configuration on the hardware micro-controller.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | **GDD\_005 >>>> CDD\_005** | | |
| Pin number | Initial direction | Functionality | Initial Value |
| Pin\_0 | Input Pulldown | Switch 0 | Low |
| Pin\_1 | Input Pulldown | Switch 1 | Low |
| Pin\_2 | Input Pulldown | Switch 2 | Low |
| Pin\_3 | Input Pulldown | Switch 3 | Low |
| Pin \_4 | Output | LCD D4 | Low |
| Pin \_5 | Output | LCD D5 | Low |
| Pin \_6 | Output | LCD D6 | Low |
| Pin \_7 | Output | LCD D7 | Low |
| Pin \_8 | Output | LCD RS | Low |
| Pin \_9 | Output | LCD RW | Low |
| Pin \_10 | Output | LCD E | Low |
| Pin\_11 | Output | Buzzer | Low |
| All Other | Output | Future use | Low |