|  |  |
| --- | --- |
|  | 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

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

**Version Control:**

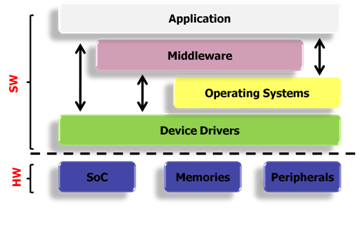
|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Modifications** |
| V 1.1 | Merna Hany  Sara Kassem  Asmaa Adel  Mahmoud Naguib  Mohamed Osama | February 12 , 2019 | First Version of the CDD, added the initial design for the software |
| V 1.2 | Merna Hany  Sara Kassem  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 | 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:**

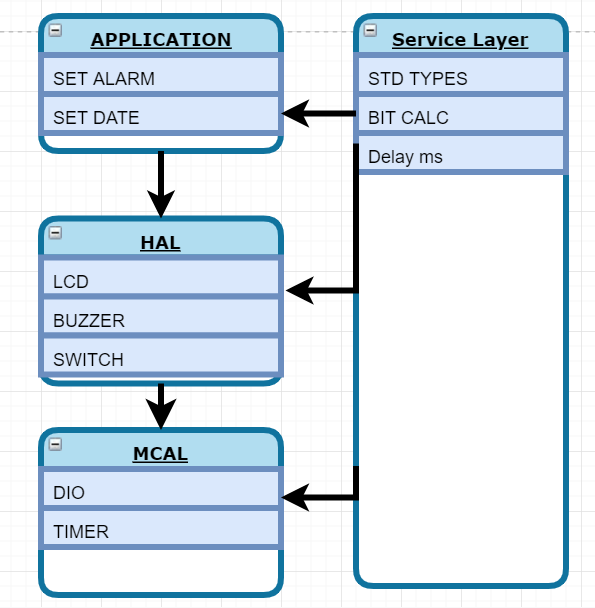
**Description:**

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. Clear\_bit

|  |  |
| --- | --- |
| **Function Name** | * Clear bit |
| **Function Prototype** | * U8 DIO\_u8ClearBit (u8 Copy\_u8PortId, u8 Copy\_u8PinId) |
| **Inputs** | * Port number * Pin number |
| **Outputs** | * Error State |
| **Description** | * The function is supposed to take the port number and the pin number to clear its value to zero. |
| **Calls** | * Nothing |
| **Coverage** | * CDD\_001 |

1. Set\_bit

|  |  |
| --- | --- |
| **Function Name** | * Set bit |
| **Function Prototype** | * U8 DIO\_u8SetBit (u8 Copy\_u8PortId, u8 Copy\_u8PinId) |
| **Inputs** | * Port number * Pin number |
| **Outputs** | * Error State |
| **Description** | The function is supposed to take the port number and the pin number to clear its value to one. |
| **Calls** | * Nothing |
| **Coverage** | * CDD\_002 |

1. Set Direction
2. Hardware Layer (HAL):
3. LCD:
4. Write\_character

|  |  |
| --- | --- |
| **Function Name** | * Write character |
| **Function Prototype** | * void LCD\_voidWriteCharacter (u8 Copy\_u8Data) |
| **Inputs** | * Data |
| **Outputs** | * Error State |
| **Description** | The function is supposed to take the data and display it on the LCD |
| **Calls** | * DIO\_SetPinValue |
| **Coverage** | * CDD\_003 |

1. Send\_command

|  |  |
| --- | --- |
| **Function Name** | * Send command |
| **Function Prototype** | * void LCD\_voidSendCommand (u8 Copy\_u8Command) |
| **Inputs** | * Command |
| **Outputs** | * Error State |
| **Description** | * The function is supposed to take the command to send to the LCD |
| **Calls** | * DIO\_SetPinValue |
| **Coverage** | * CDD\_004 |

1. Write\_character

|  |  |
| --- | --- |
| **Function Name** | * Write character |
| **Function Prototype** | * void LCD\_voidWriteCharacter (u8 Copy\_u8Data) |
| **Inputs** | * Data |
| **Outputs** | * Error State |
| **Description** | The function is supposed to take the data and display it on the LCD |
| **Calls** | * DIO\_SetPinValue |
| **Coverage** | * CDD\_005 |

1. Push Button
2. Is\_pressed

|  |  |
| --- | --- |
| **Function Name** | * Is pressed |
| **Function Prototype** | u8 BUTTON\_u8IsPressed(u8 Copy\_u8PinId) |
| **Inputs** | * status of the button * The number of the connected pin |
| **Outputs** | * Error State |
| **Description** | The function is supposed to check if the button is pressed or not |
| **Calls** | * DIO\_GetPinValue |
| **Coverage** | * CDD\_006 |

1. Buzzer
2. Start\_Buzzer

|  |  |
| --- | --- |
| **Function Name** | * Start |
| **Function Prototype** | u8 BUZZER\_u8Start (u8 Copy\_u8PinId) |
| **Inputs** | * The number of the connected pin |
| **Outputs** | * Error State |
| **Description** | The function is supposed to start the buzzer |
| **Calls** | * DIO\_SetPinValue |
| **Coverage** | * CDD\_007 |

1. Stop\_Buzzer

|  |  |
| --- | --- |
| **Function Name** | * Stop |
| **Function Prototype** | u8 BUZZER\_u8Stop(u8 Copy\_u8PinId) |
| **Inputs** | * The number of the connected pin |
| **Outputs** | * Error State |
| **Description** | The function is supposed to stop the buzzer |
| **Calls** | * DIO\_SetPinValue |
| **Coverage** | * CDD\_007 |

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\_008 |

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\_009 |

**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 |