DESIGN DOCUMENT

AIR CONDITIONER

Team7:

-Amr El-Abd

-Kareem Magdy

-Nada Abdelazim

-Moustafa Abdelrahim

Table of Contents: -

Subject **Page**

Project introduction:

The aim of this project is to develop a system that controls the temperature of a specific environment. The system includes a temperature sensor, a keypad, and a buzzer. The user can set the desired temperature range using the keypad. When the temperature exceeds the set range, the buzzer will activate, warning the user about the temperature increase.

Project description :-

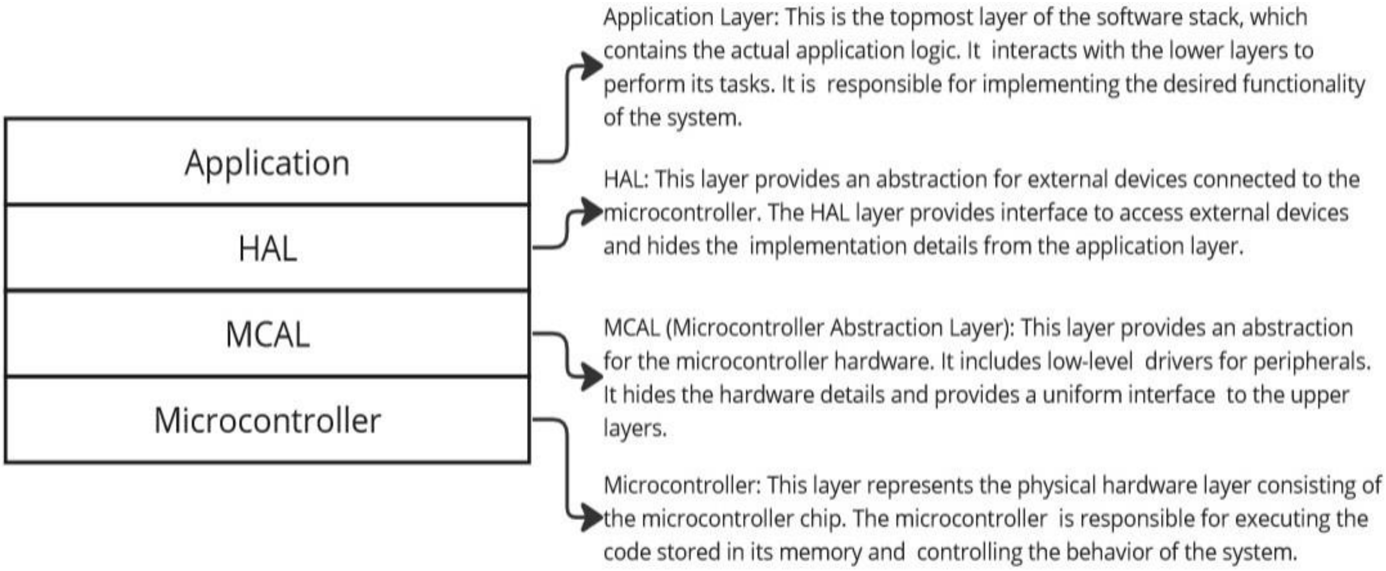
The system's functionality can be summarized in the following points:

1. Upon powering on, the system will display a welcome message on the LCD for one second, then clear the display.
2. The system will display the minimum and maximum temperature values which are 18 C and 35 C respectively for half a second.
3. The system will ask the user to choose the required temperature by displaying "Please Choose Required Temp" on the LCD for half a second.
4. The user can increase or decrease the temperature range using the KEYPAD 1 and KEYPAD 2 buttons, respectively.
5. The user can set the desired temperature range using the KEYPAD 3 button.
6. If the user presses KEYPAD 1, KEYPAD 2, or KEYPAD 3 after setting the temperature range, the system will display "This Operation Is Not Allowed" for half a second.
7. The system will continuously display the current temperature on the LCD.
8. If the current temperature exceeds the set temperature range, the system will activate the buzzer to alert the user and print a bell shape on the LCD.
9. The user can stop the buzzer using the KEYPAD 4 button.
10. The user can reset the temperature to the default value (20 C) using the KEYPAD 5 button.

Hardware components:

- LCD
- Keypad
- Temperature sensor
- Buzzer
- Software

Layered Architectures:-



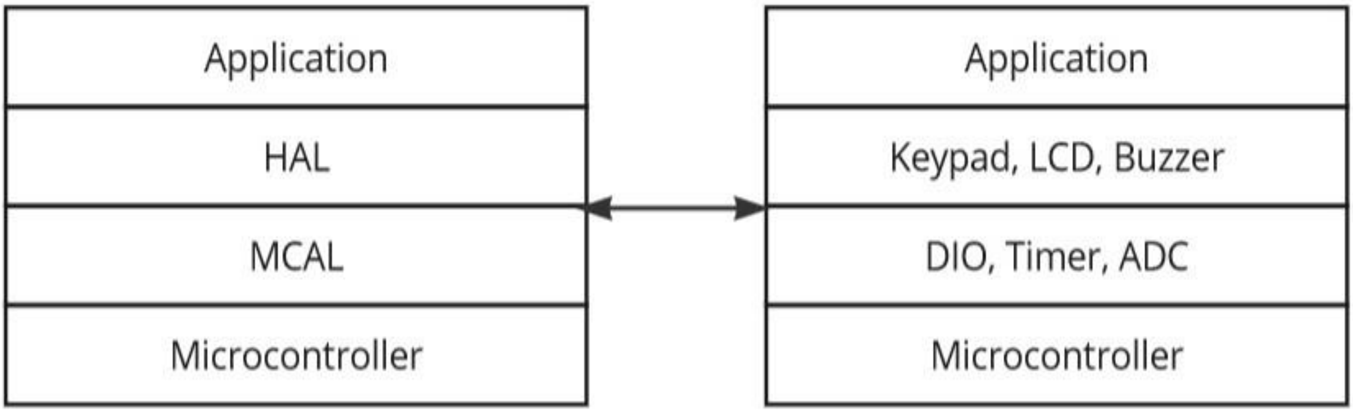
Application Layer: This is the topmost layer of the software stack, which contains the actual application logic. It interacts with the lower layers to perform its tasks. It is responsible for implementing the desired functionality of the system.

HAL: This layer provides an abstraction for external devices connected to the microcontroller. The HAL layer provides interface to access external devices and hides the implementation details from the application layer.

MCAL (Microcontroller Abstraction Layer): This layer provides an abstraction for the microcontroller hardware. It includes low-level drivers for peripherals. It hides the hardware details and provides a uniform interface to the upper layers.

Microcontroller: This layer represents the physical hardware layer consisting of the microcontroller chip. The microcontroller is responsible for executing the code stored in its memory and controlling the behavior of the system.

System modules:-



Drivers' documentation:-

HAL drivers:

1. LCD Driver:

Description: This driver controls the LCD display and provides an interface between the microcontroller and the LCD hardware, allowing the microcontroller to display the temperature readings and messages to the user.

Functions:

```
void LCD_WRITE_COMMAND(uint8_t a_COMMAND);
void LCD_WRITE_DATA(uint8_t a_DATA);
void LCD_INIT(void);
void LCD_Write_String(uint8_t*a_String);
void LCD_Write_Number(uint32_t a_number);
void LCD_Clear(void);
void LCD_GoTo(uint8_t a_line,uint8_t a_cell);
void LCD_Write_Charecter(uint8_t a_char);
```

2. Keypad Driver:

Description: This driver provides an interface between the microcontroller and the keypad hardware, allowing the microcontroller to receive input from the user through the keypad buttons.

Functions:

```
void KEYPAD_init(void);
uint8_t KEYPAD_getKey(void);
```

3. Buzzer Driver:

Description: This driver controls the buzzer and provides an interface between the microcontroller and the buzzer hardware, allowing the microcontroller to activate and deactivate the buzzer to alert the user when the temperature exceeds the set range.

Functions:

```
void buzzer_init(void);
void buzzer_On(void);
void buzzer_Off(void);
```

MCAL drivers:

4. DIO Driver:

Description: The DIO (Digital Input Output) driver is responsible for setting up the digital pins of the microcontroller to either input or output mode. This driver will be used to control the buttons and LEDs.

Functions:

```
DIO_ERROR_TYPE DIO_INITPIN(DIO_PIN_TYPE PIN,DIO_PINSTATUS_TYPE STATUS);
DIO_ERROR_TYPE DIO_WRITEPIN(DIO_PIN_TYPE PIN,DIO_VOLTAGE_TYPE VOLTAGE);
DIO_ERROR_TYPE DIO_READPIN(DIO_PIN_TYPE PIN,DIO_VOLTAGE_TYPE* VOLT);
void DIO_TogglePin(DIO_PIN_TYPE pin);
```

5. Timer Driver:

Description: The Timer driver is responsible for setting up and controlling the timers of the microcontroller. This driver will be used to create the timing delays required in the project.

Functions:

```
//timer 0 prototypes

Timer_ErrorStatus TIMER_0_init(Timer_Mode mode);
Timer_ErrorStatus TIMER_0_start(Timer_Prescaler prescaler);
void TIMER_0_stop(void);
Timer_ErrorStatus TIMER_0_setIntialValue(uint8_t value);
Timer_ErrorStatus TIMER_0_OvfNum(double overflow);
void TIMER_0_DELAY_MS(double _delay);

//timer 2 prototypes

Timer_ErrorStatus TIMER_2_init(Timer_Mode mode);
Timer_ErrorStatus TIMER_2_start(Timer_Prescaler prescaler);
void TIMER_2_stop(void);
Timer_ErrorStatus TIMER_2_setIntialValue(uint8_t value);
Timer_ErrorStatus TIMER_2_OvfNum(double overflow);
void TIMER_2_DELAY_MS(double _delay);
void TIMER_2_INT();

//PWM Function prototype

void TIMER_0_pwm(float intial);
```

6. ADC Driver :

Description: This driver controls the Analog to Digital Converter (ADC) and provides an interface between the microcontroller and the temperature sensor hardware, allowing the microcontroller to convert the analog temperature signal to a digital value that can be processed and maintained within the temperature control system.

Functions:

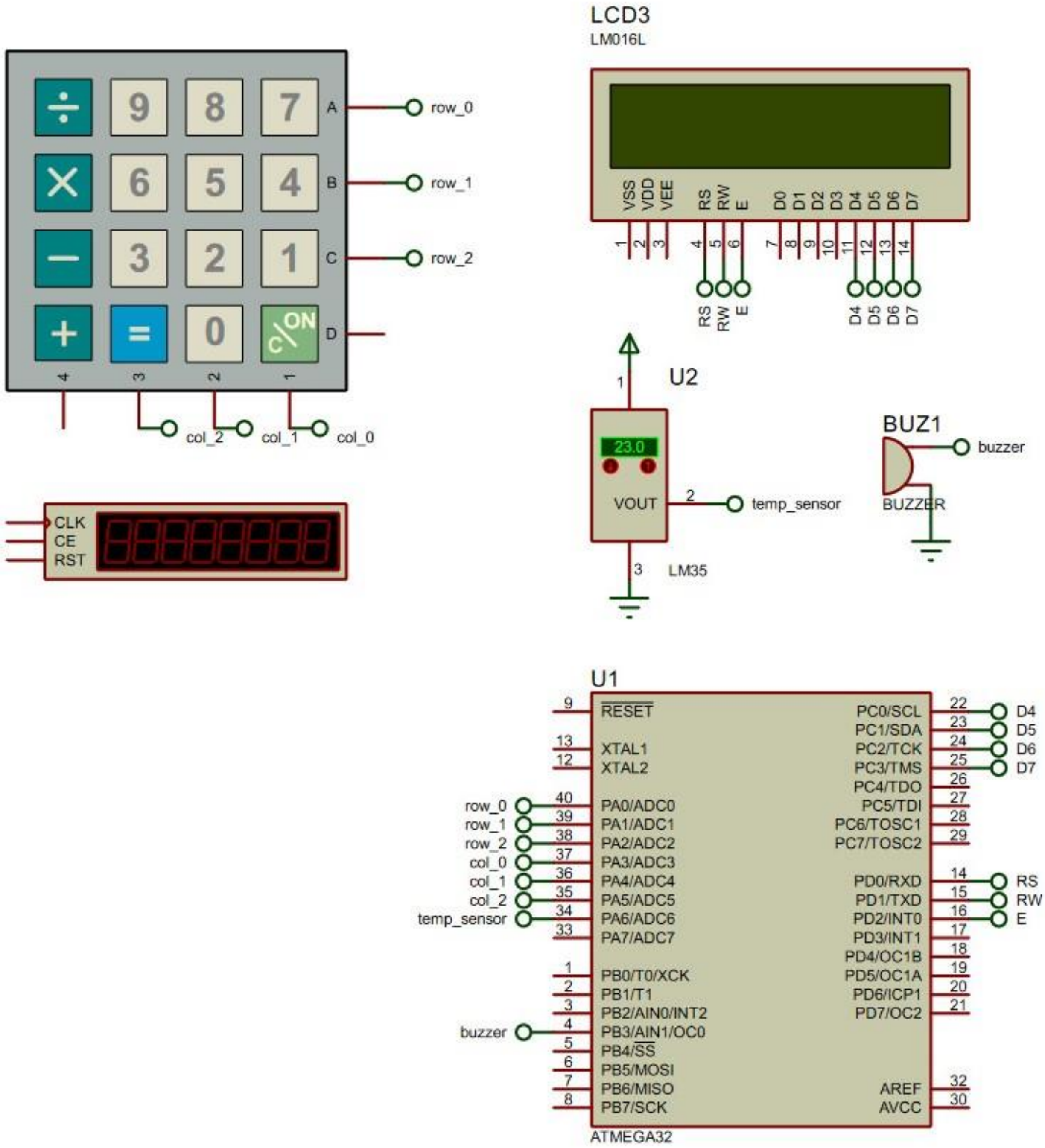
```
void ADC_init(void);

void ADC_start_conversion (ADC_CH_type ADC_CH);

uint16 ADC_Read(void);

uint16 ADC_LM35_calibration (void);
```

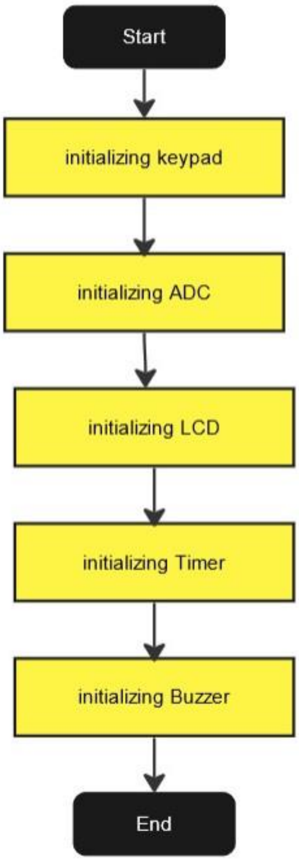
Proteus simulation design :-



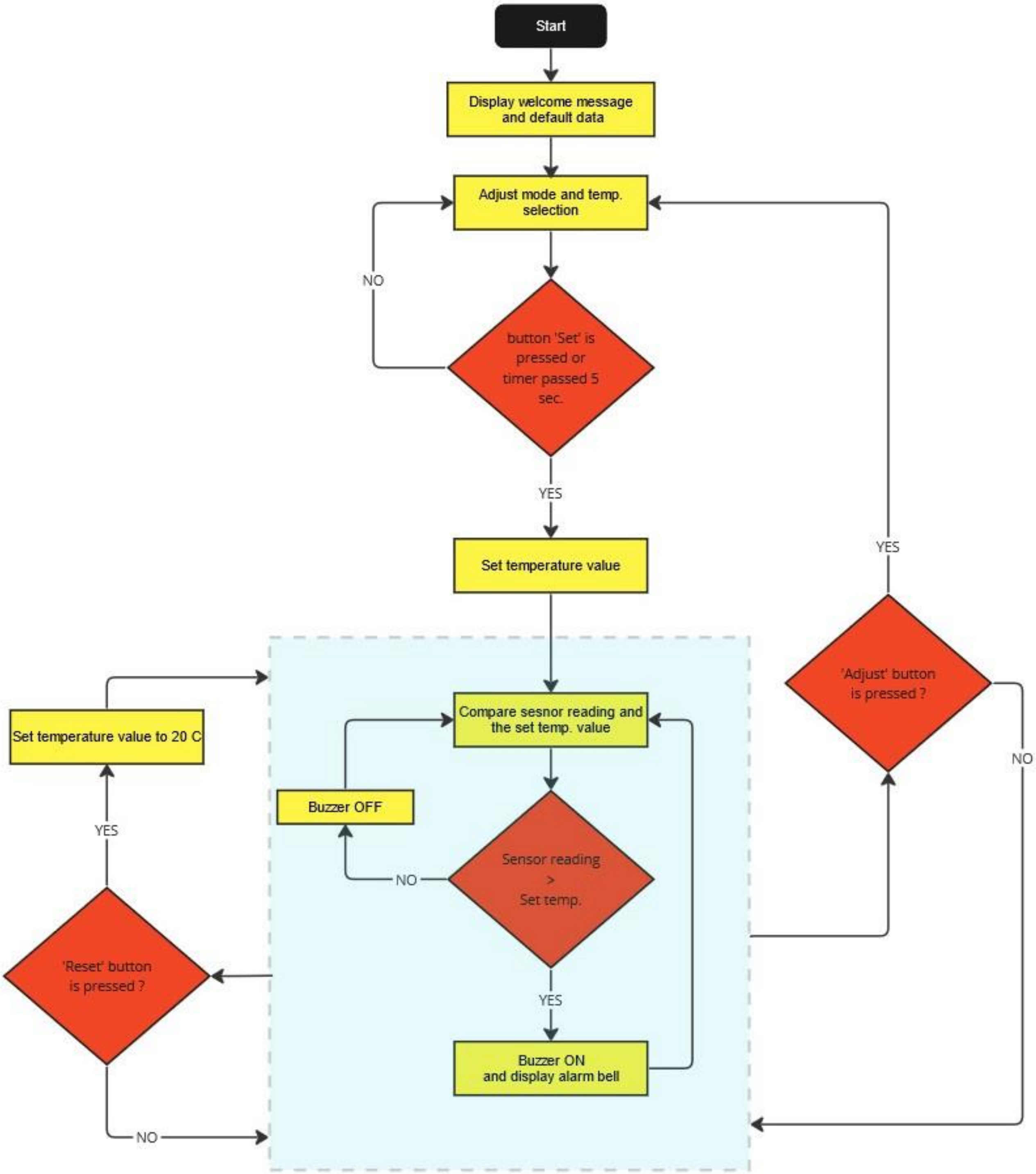
Flowcharts for Functions from Higher layers downwards:

App layer functions flowcharts:

```
void app_Init(void);
```



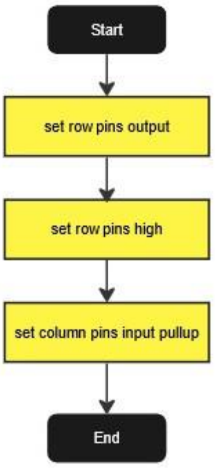
```
void app_Start(void);
```


Start
Display welcome message and default data
Adjust mode and temp. selection
button 'Set' is pressed or timer passed 5 sec.
NO
YES
Set temperature value
Compare sesnor reading and the set temp. value
Sensor reading > Set temp.
NO
Buzzer OFF
YES
Buzzer ON and display alarm bell
'Adjust' button is pressed ?
YES
NO
'Reset' button is pressed ?
YES
Set temperature value to 20 C
NO

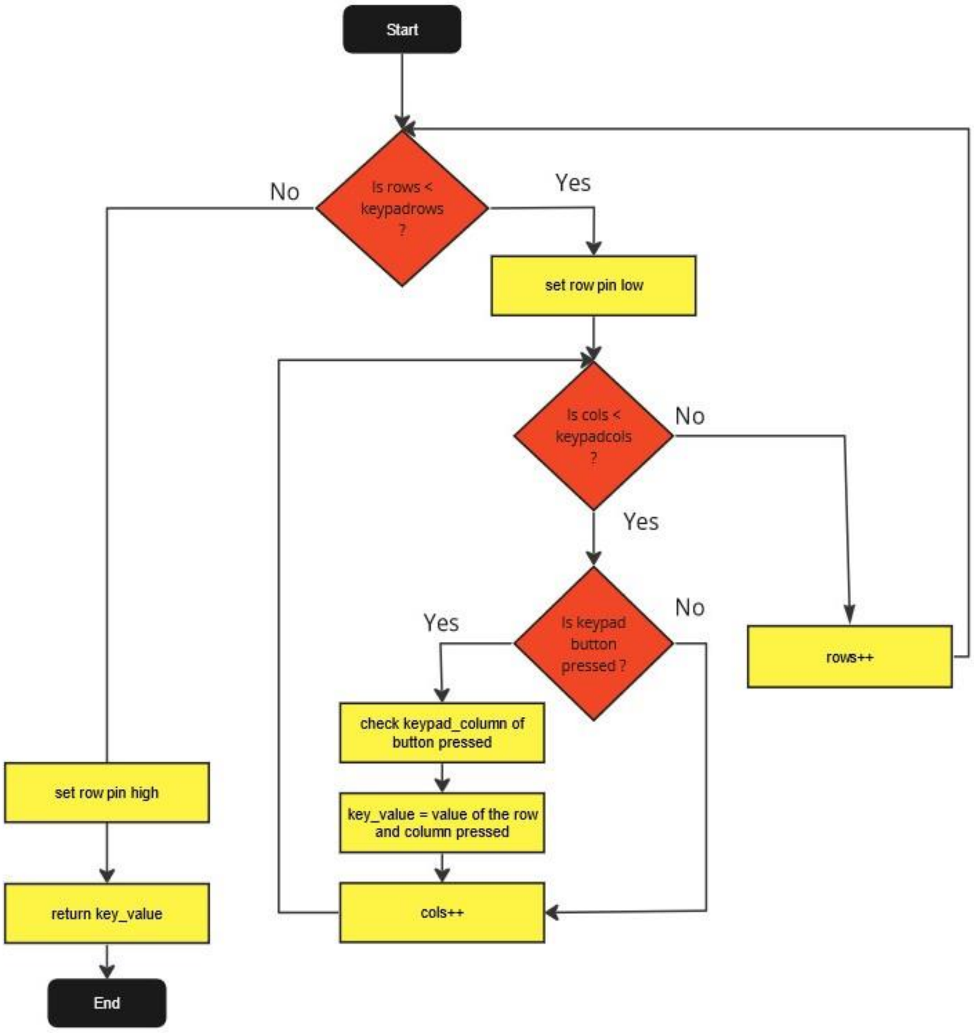
HAL Layer:

Keypad functions:-

```
void KEYPAD_init(void);
```

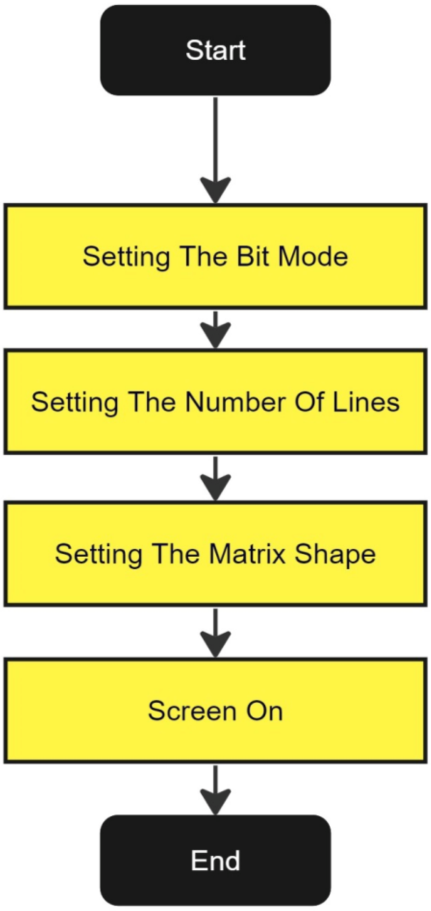


```
uint8_t KEYPAD_getKey(void);
```

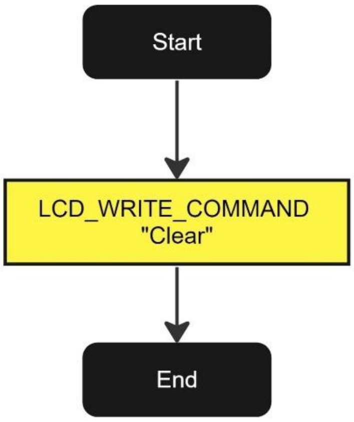


LCD functions:-

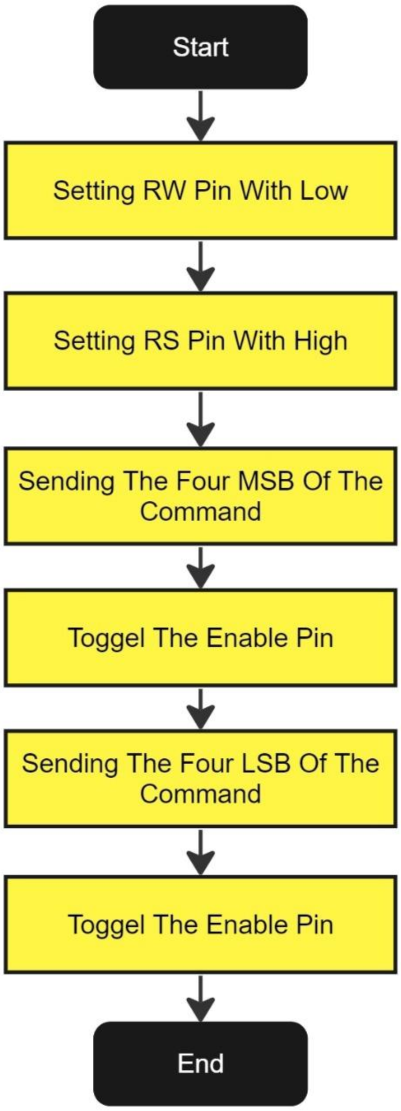
LCD_INIT



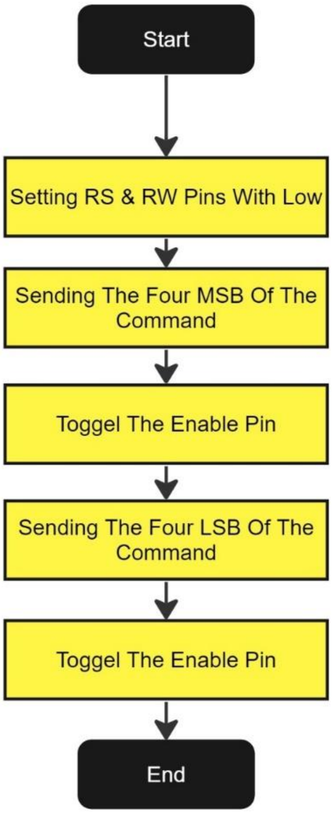
LCD_CLEAR



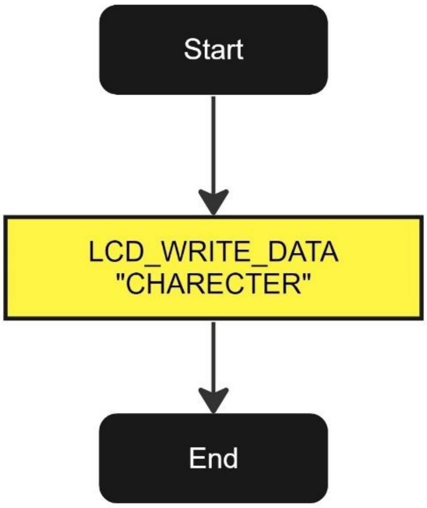
LCD_WRITE_DATA



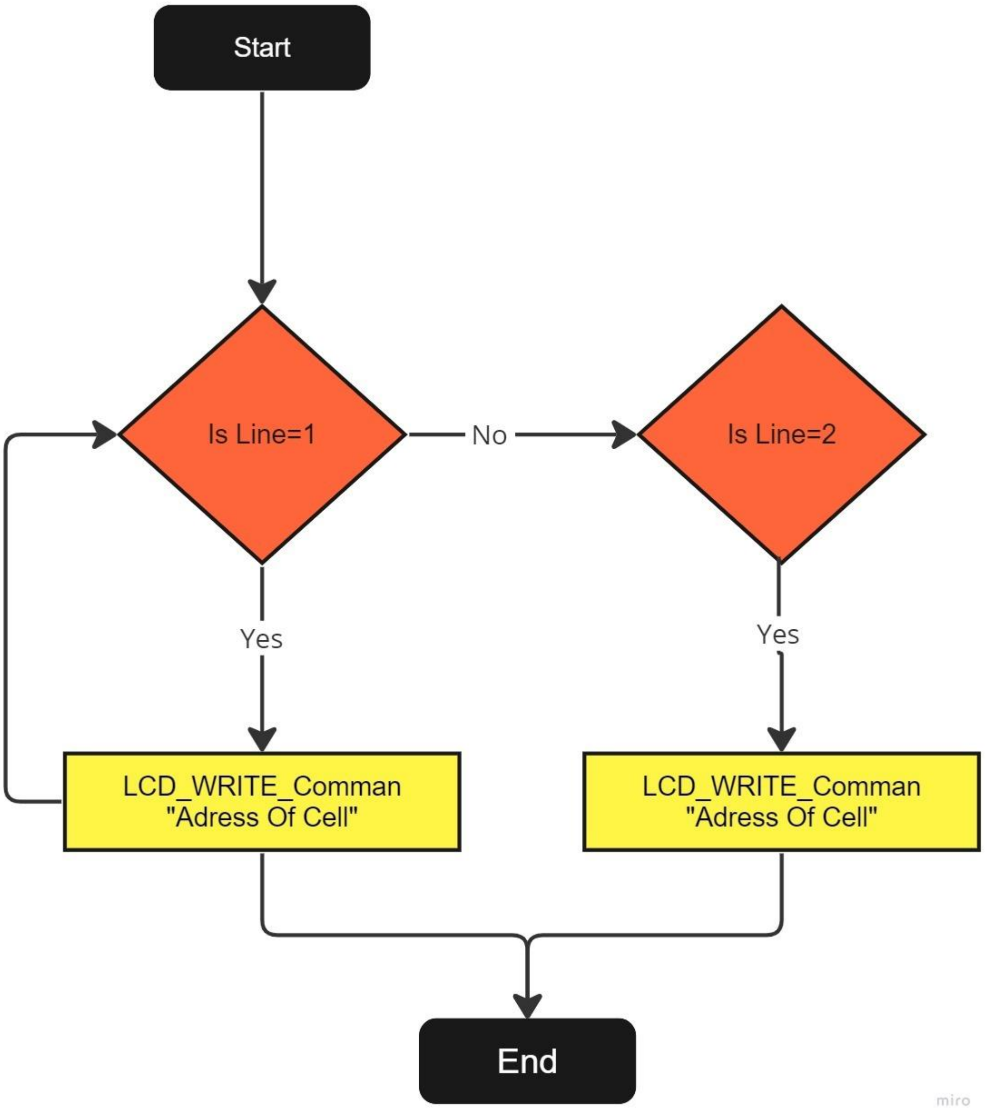
LCD_WRITE_COMMAND


Start
Setting RS & RW Pins With Low
Sending The Four MSB Of The Command
Toggel The Enable Pin
Sending The Four LSB Of The Command
Toggel The Enable Pin
End

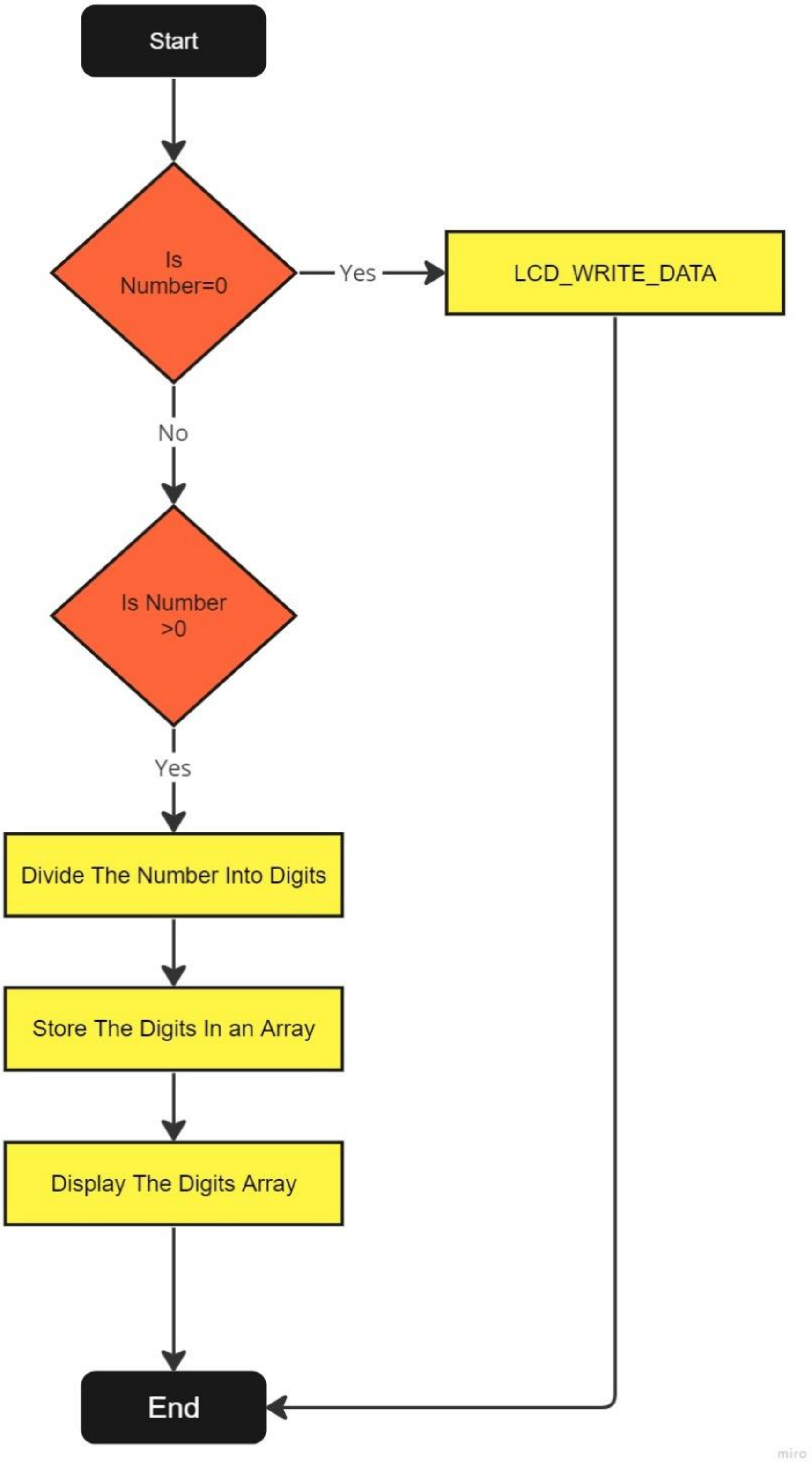
LCD_Write_Charecter



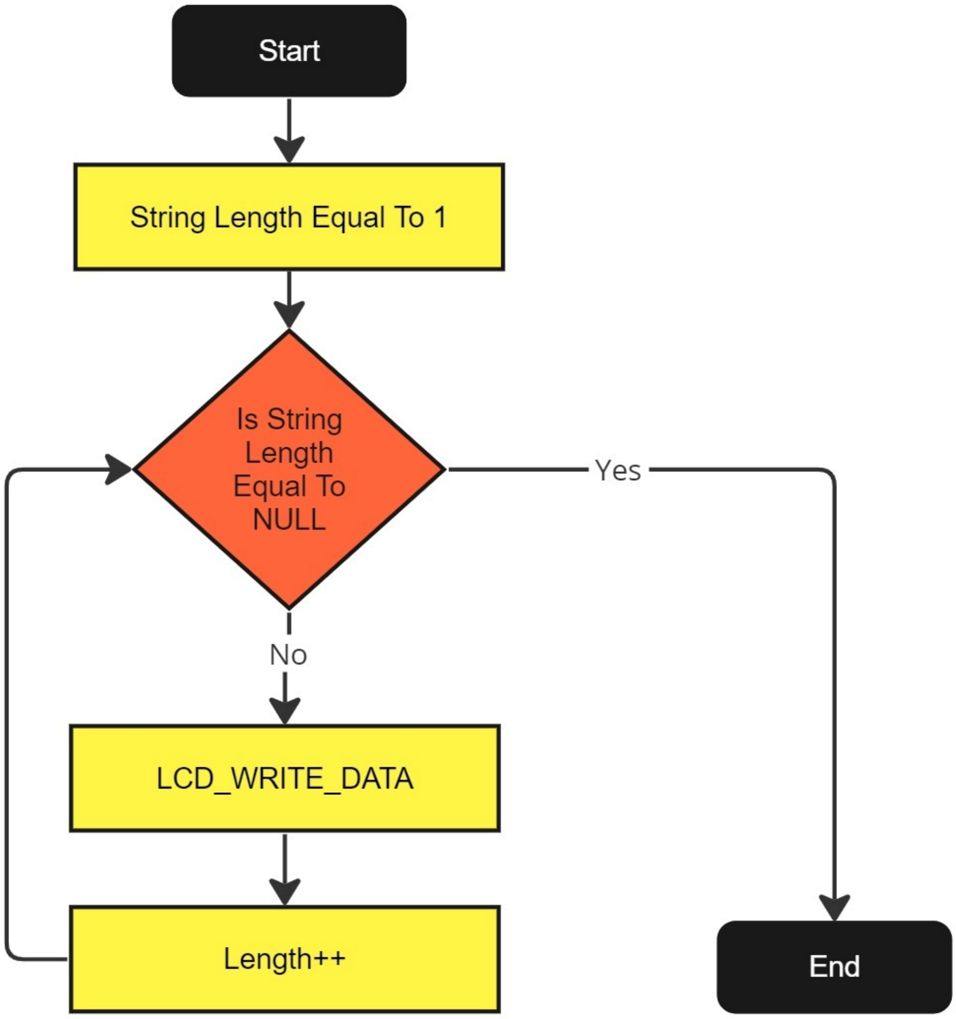
LCD_GoTo

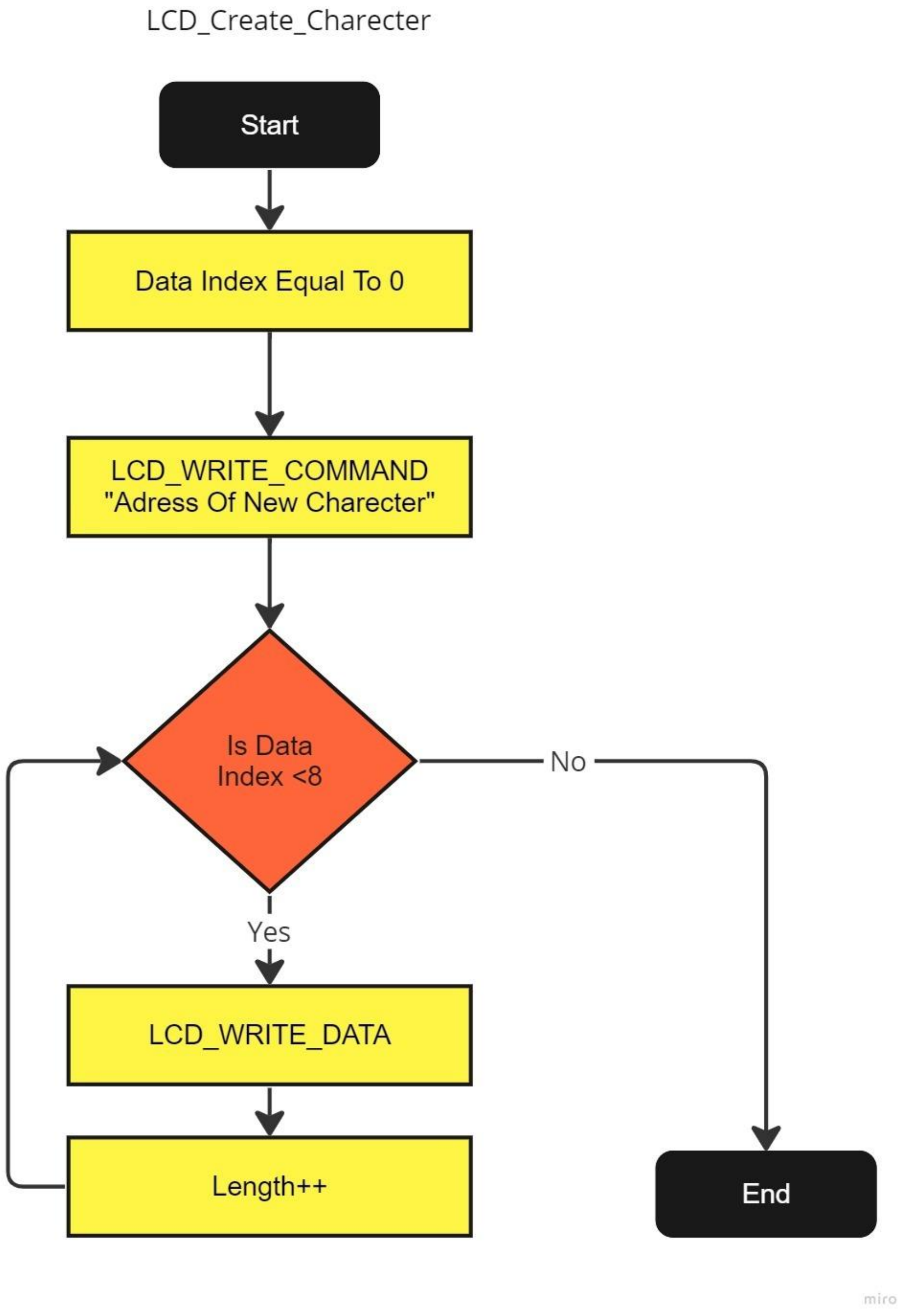

Start
Is Line=1
No
Is Line=2
Yes
Yes
LCD_WRITE_Comman "Adress Of Cell"
LCD_WRITE_Comman "Adress Of Cell"
End
miro

LCD_WRITE_NUMBER



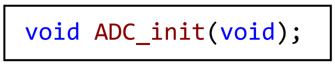
LCD_Write_String



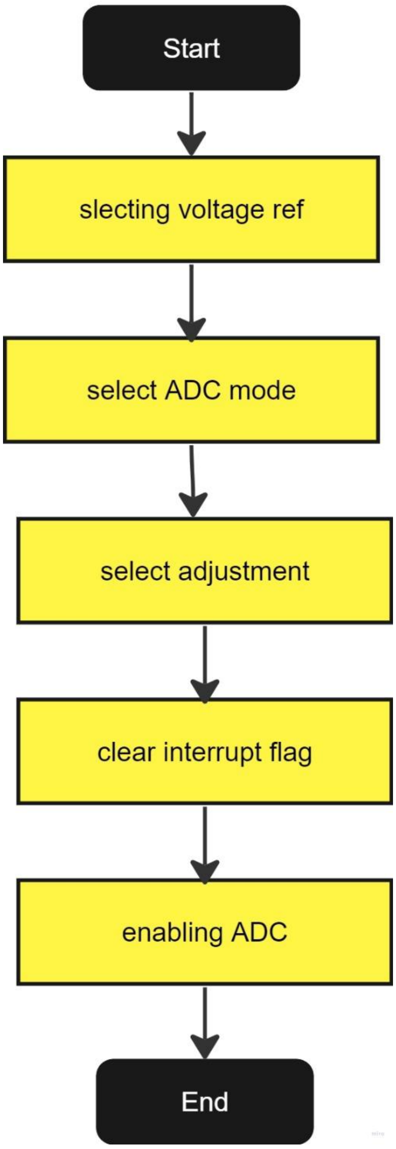
LCD_Create_Charecter
Start
Data Index Equal To 0
LCD_WRITE_COMMAND
"Adress Of New Charecter"
Is Data
Index <8
No
Yes
LCD_WRITE_DATA
Length++
End
miro

MCAL Layer:-

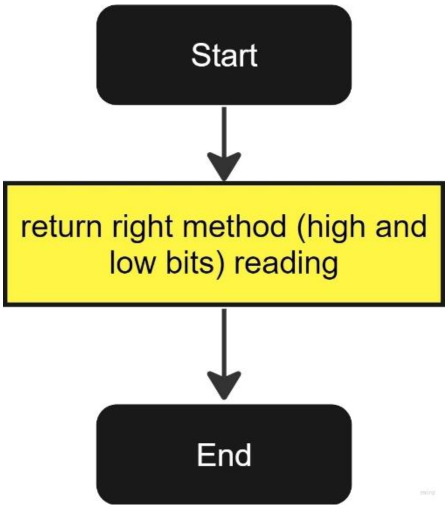
ADC Functions :-



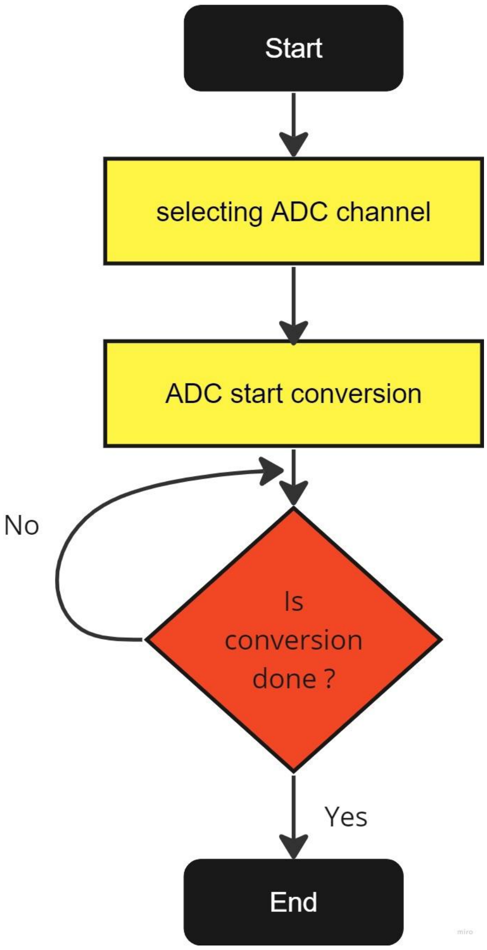
```
void ADC_init(void);
```



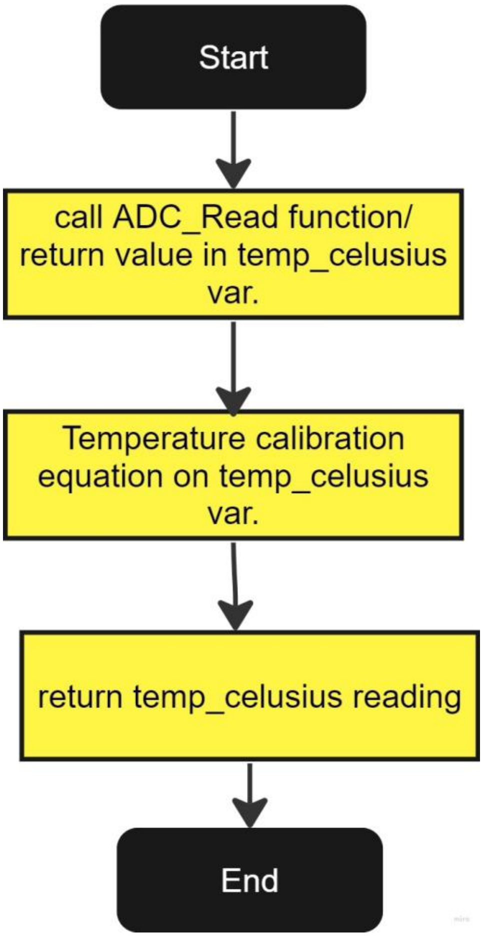
```
uint16 ADC_Read(void);
```



```
void ADC_start_conversion (ADC_CH_type ADC_CH);
```

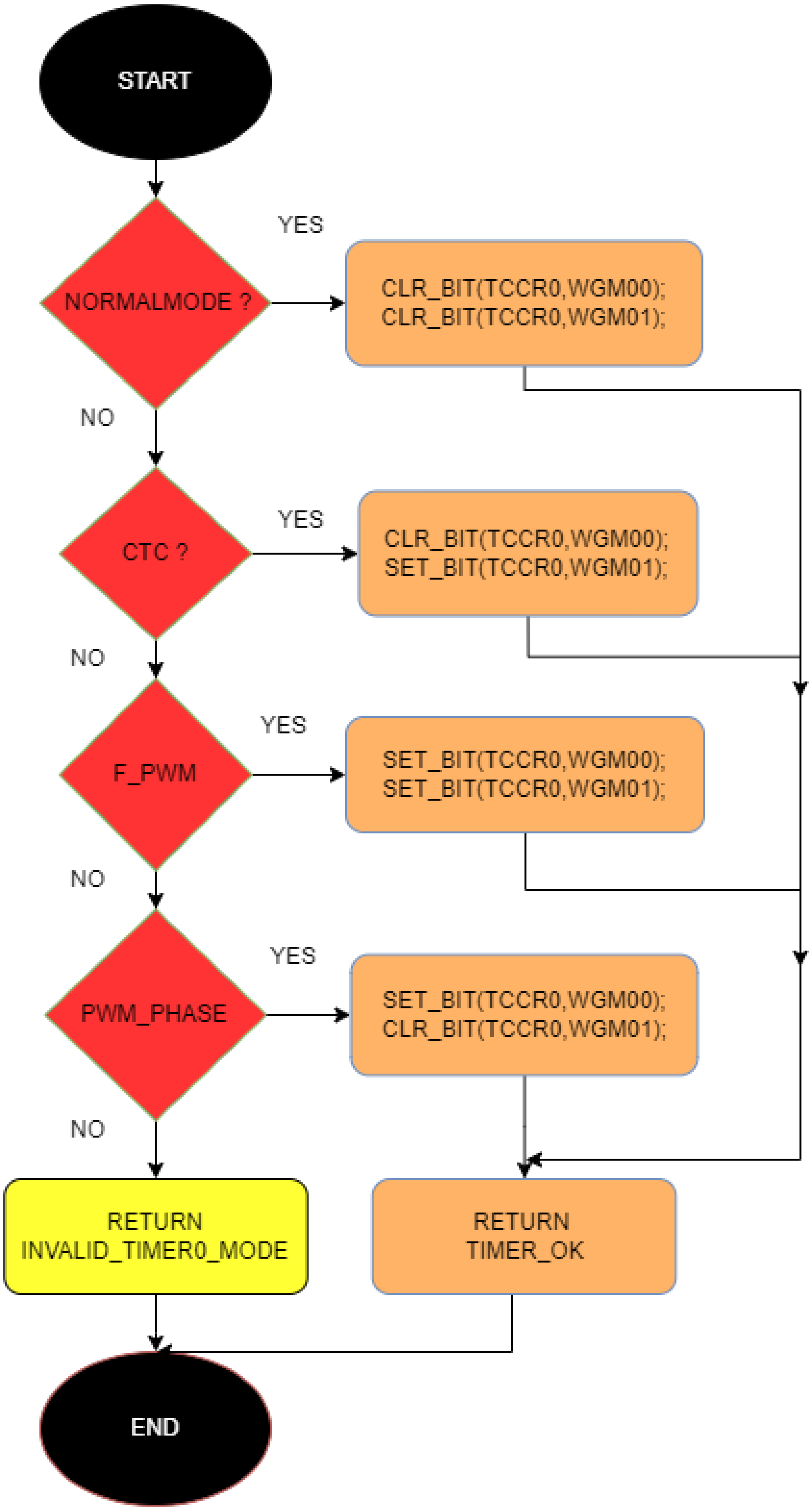


```
uint16 ADC_LM35_calibration (void);
```

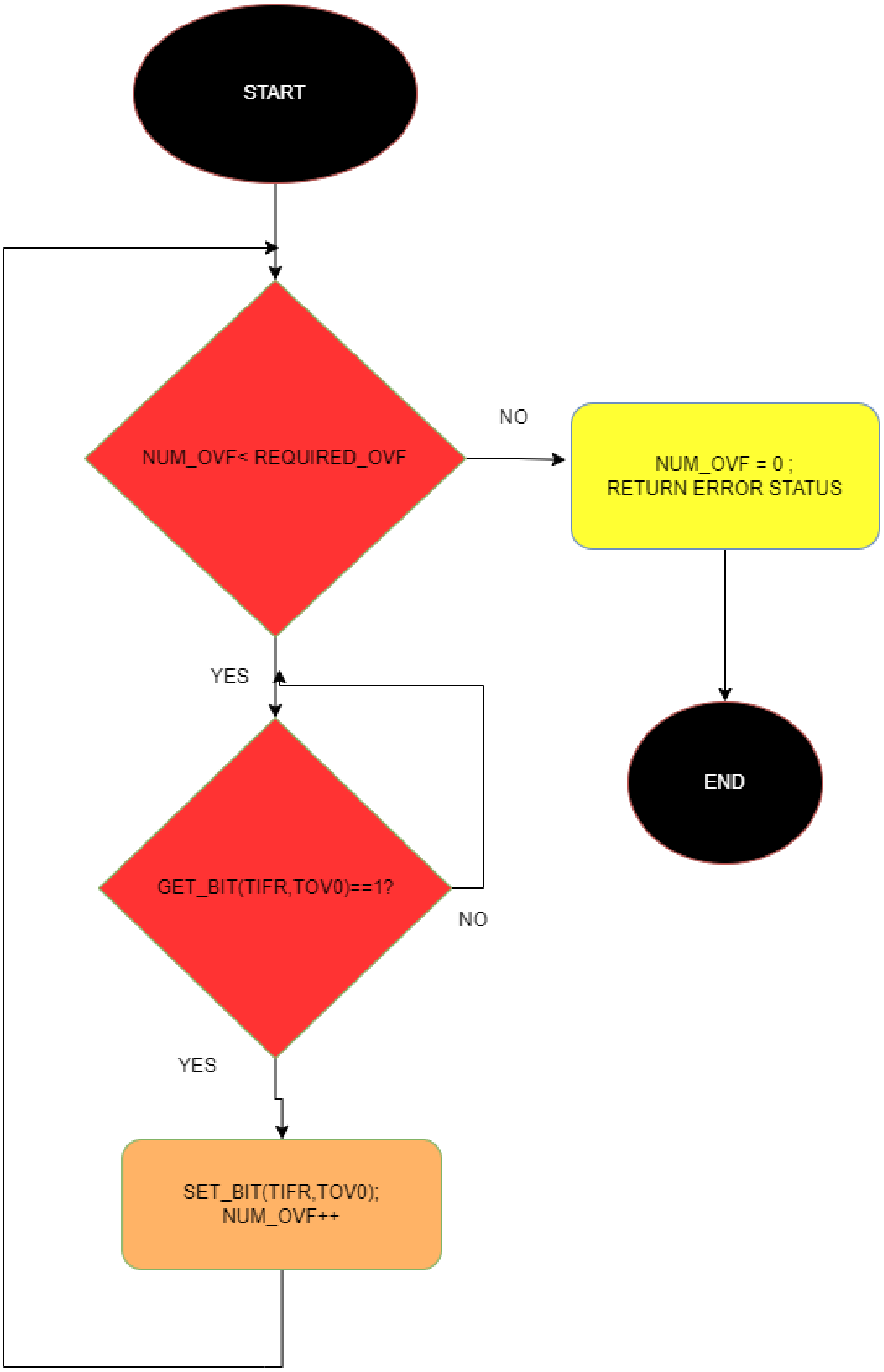

Start
call ADC_Read function/ return value in temp_celusius var.
Temperature calibration equation on temp_celusius var.
return temp_celusius reading
End

Timer functions’ flowcharts:-

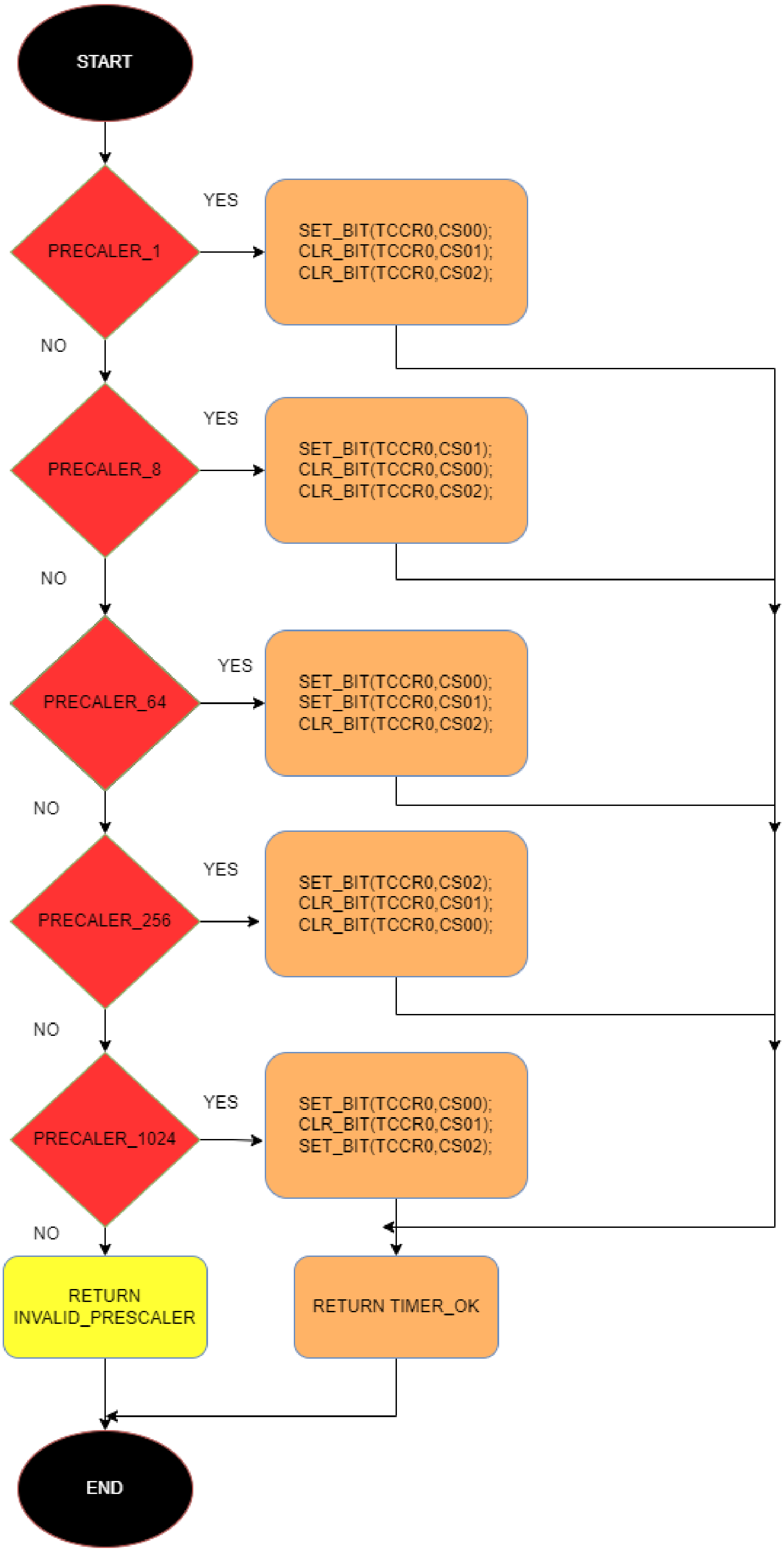
```
Timer_ErrorStatus TIMER_0_init(Timer_Mode mode);
```

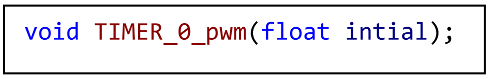


```
Timer_ErrorStatus TIMER_0_OvfNum(double overflow);
```

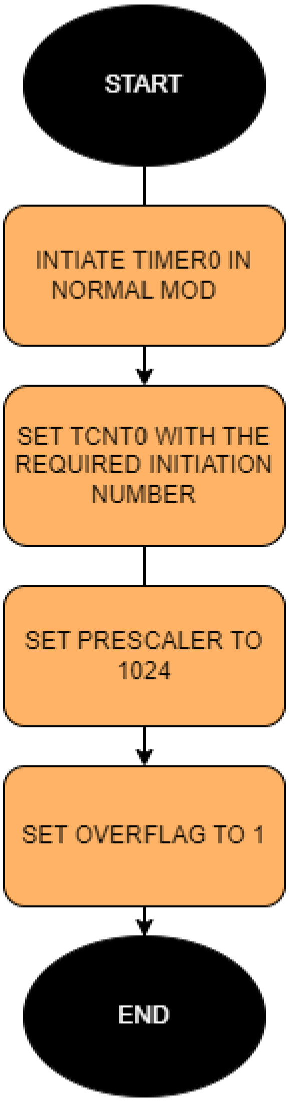


```
Timer_ErrorStatus TIMER_0_start(Timer_Prescaler prescaler);
```



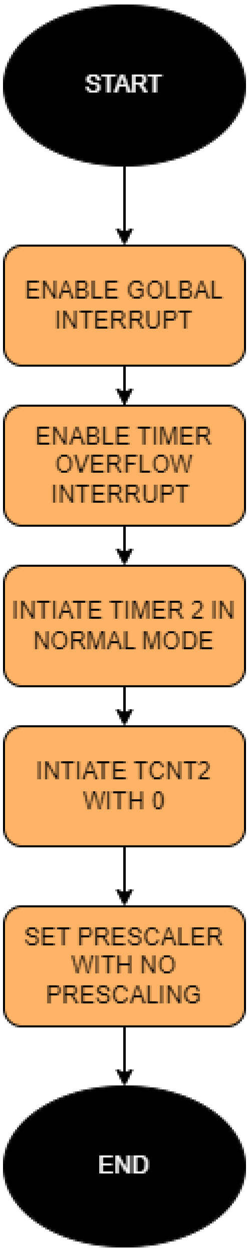


```
void TIMER_0_pwm(float intial);
```



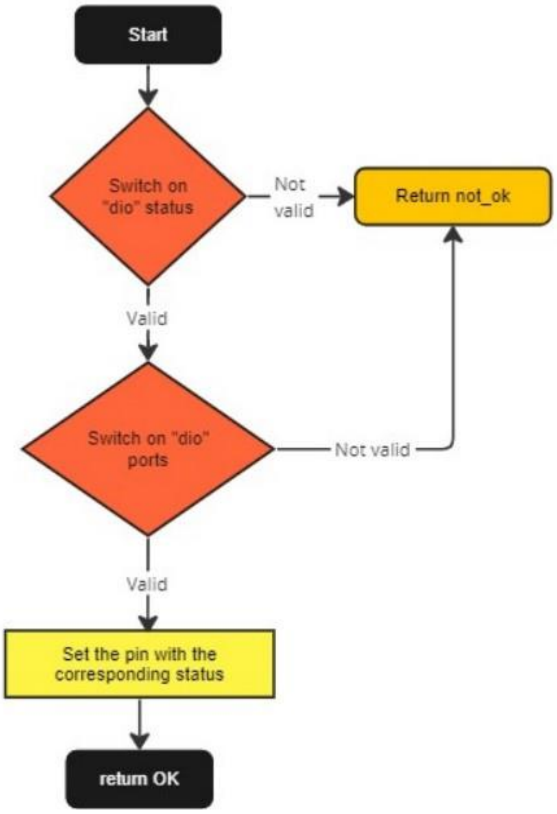
START
INTIATE TIMER0 IN NORMAL MOD
SET TCNT0 WITH THE REQUIRED INITIATION NUMBER
SET PRESCALER TO 1024
SET OVERFLAG TO 1
END

TIMER 2 WITH INTERRUPT

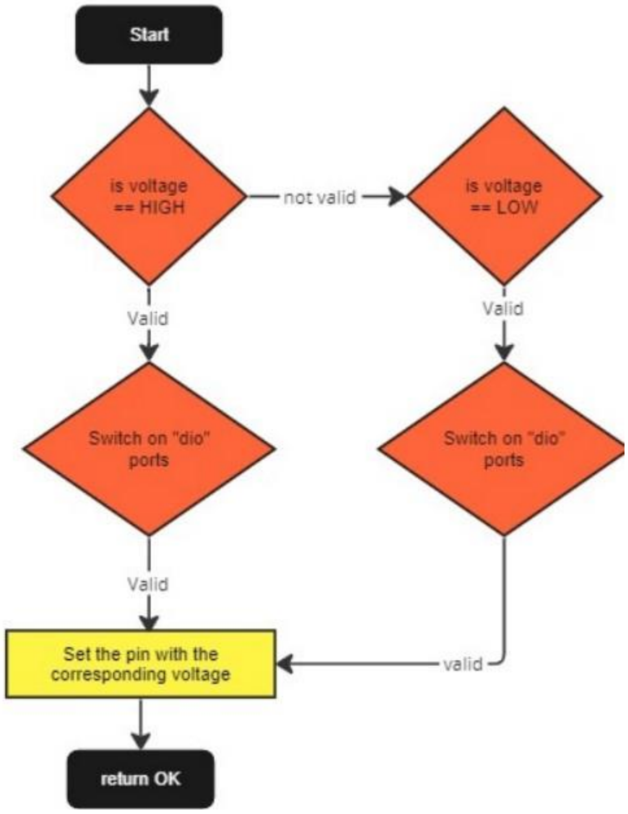

START
ENABLE GOLBAL INTERRUPT
ENABLE TIMER OVERFLOW INTERRUPT
INTIATE TIMER 2 IN NORMAL MODE
INTIATE TCNT2 WITH 0
SET PRESCALER WITH NO PRESCALING
END

DIO functions flowcharts:-

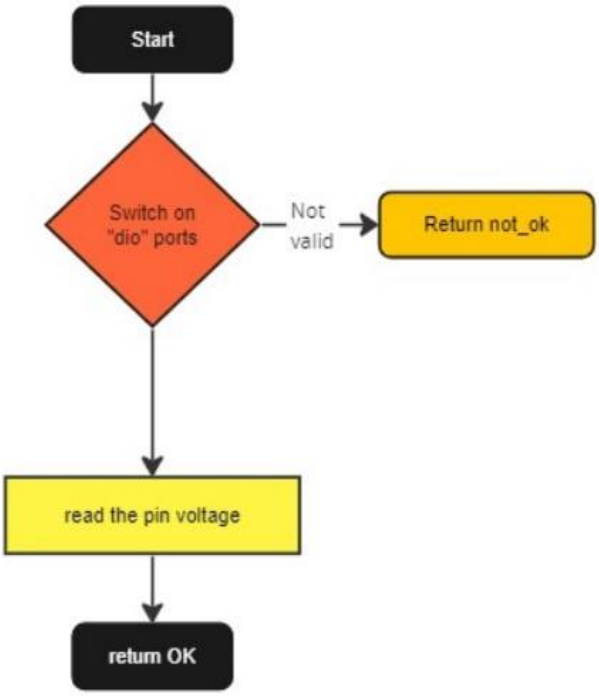
DIO_INITPIN



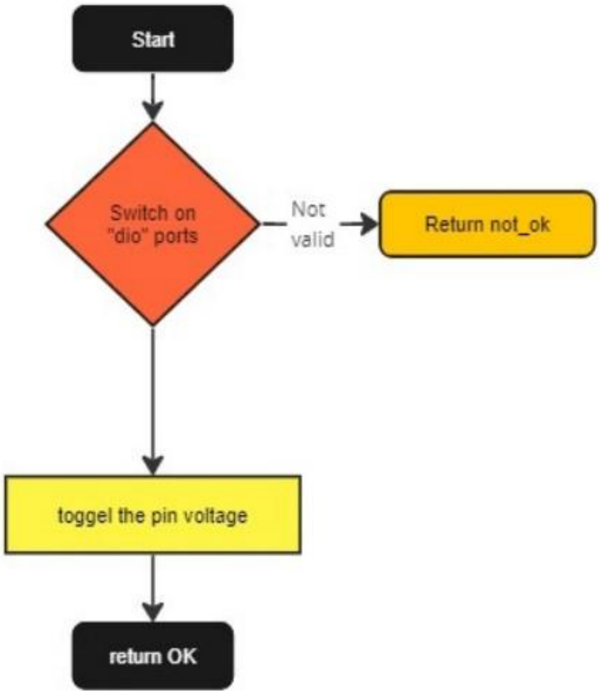
DIO_WRITEPIN



DIO_READPIN



DIO_TogglePin


Start
Switch on "dio" ports
Not valid
Return not_ok
toggel the pin voltage
return OK