DESIGN DOCUMENT

OBSTACLE AVOIDANCE CAR DESIGN

Prepared By:

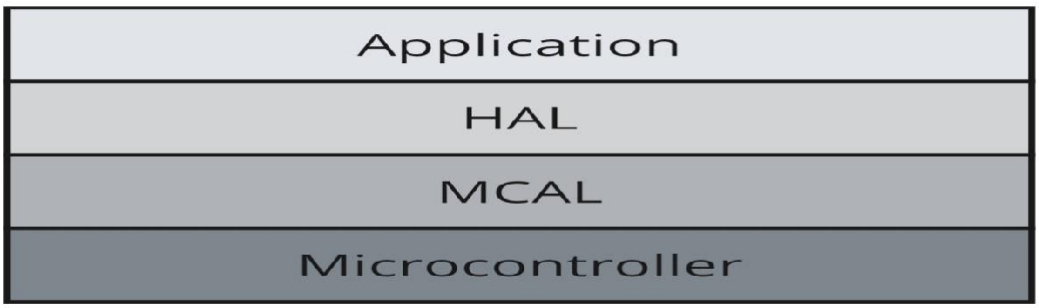
Moustafa Abd-Elrahim

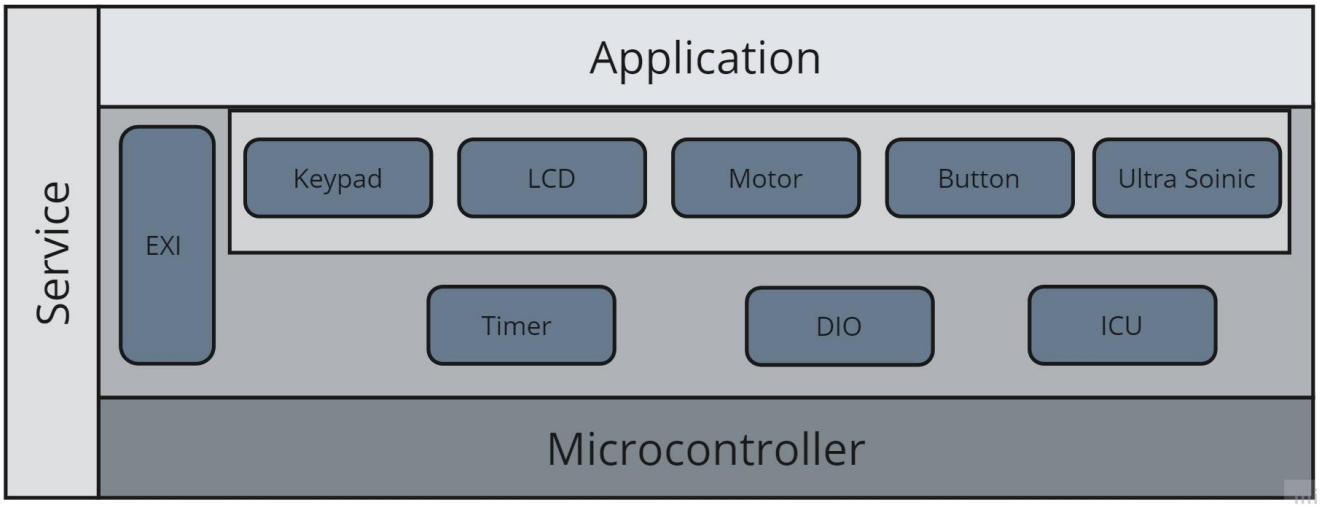
1. Project introduction:

The aim of this project is to design a four wheels driving car that has the ability to avoid any type of obstacle with a self-enhancing mechanism that changes speed and direction according to the circumstances

2. High Level Design:

a. Layered Architectures:





b. Modules description:

i. **MCAL:** *It 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.*

ii. **HAL:** *It is a layer of programming that allows The Application Layer to interact with a hardware device at a general or abstract level rather than at a detailed hardware level.*

iii. **Application:** *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.*

c. Drivers' documentation:-

i. 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);
```

ii. 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);
```

iii. Interrupt Driver:

Description: *The Interrupt driver is responsible for setting up and controlling the interrupts of the microcontroller. This driver will be used to detect button presses.*

Functions:

```
EN_int__error_t EXI_Enable (EN_int_t Interrupt);
EN_int__error_t EXI_Disable (EN_int_t Interrupt);
EN_int__error_t EXI_Trigger(EN_int_t Interrupt,EN_trig trigger);
void EXI_SetCallBack(EN_int_t Interrupt,void(*ptrf)(void));
```

iv. Button Driver:

Description: *The Button driver is responsible for setting up and controlling the buttons of the microcontroller. This driver will be used to detect button presses.*

Functions:

```
BUTTON_ERROR_TYPE Button_INIT(DIO_PIN_TYPE PIN);
BUTTON_ERROR_TYPE Button_read(DIO_PIN_TYPE PIN,DIO_VOLTAGE_TYPE*VOLT);
```

v.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);
```

vi. Motor Driver:

Description: *The Motor driver is responsible for setting up and controlling the motors of the car. This driver will be used to control the speed and direction of the motors.*

Functions:

```
void Motors_init(void);
void Motors_Start(void);
void Motors_Rotating(void);
void Motors_Stop(void);
```

VII. ICU Driver:

Description: *the ICU module is used to capture a timer value from one of two selectable time bases on the occurrence of an event on an input pin.*

Functions:

```
void ICU_init(void);
```

VIII. Ultra Sonic Driver:

Description: *An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal*

Function:

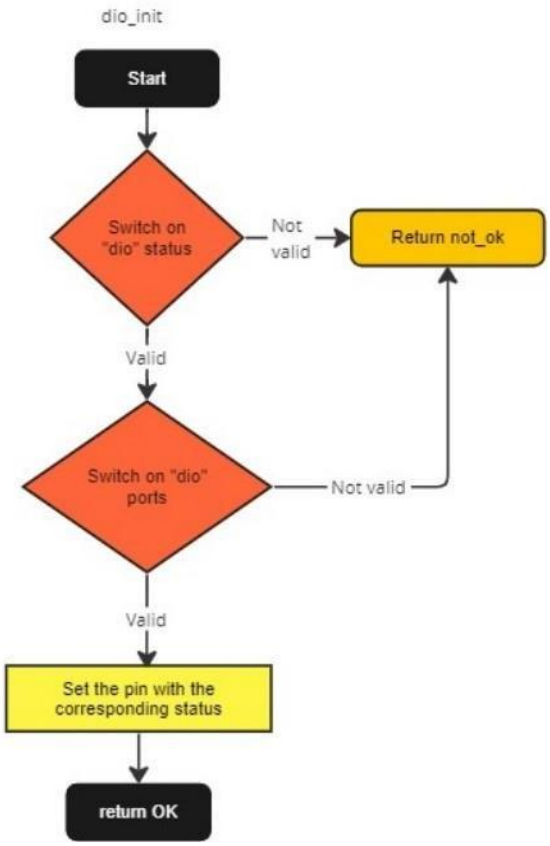
```
void US_init(void);
void US_GetDistance(void);
```

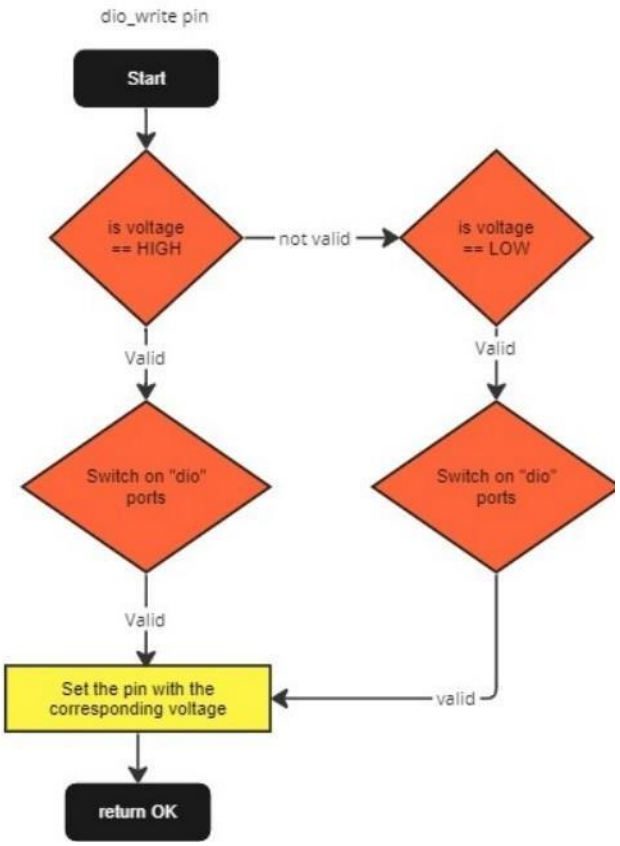
3. Low Level Design:

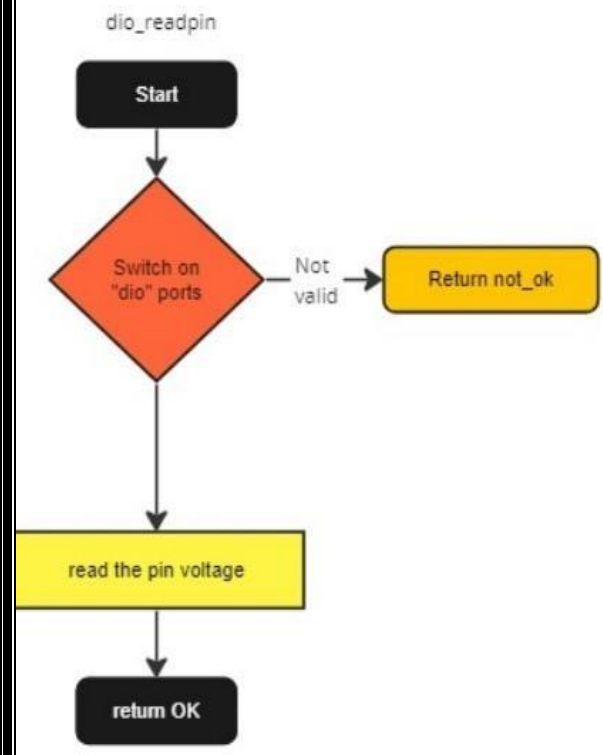
a. Flow Charts :

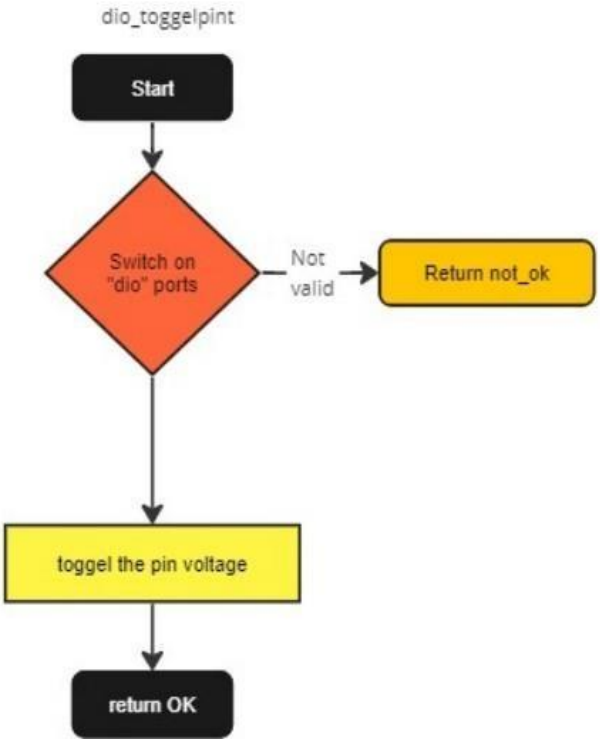
1. MCAL Layer:

i. DIO Functions:

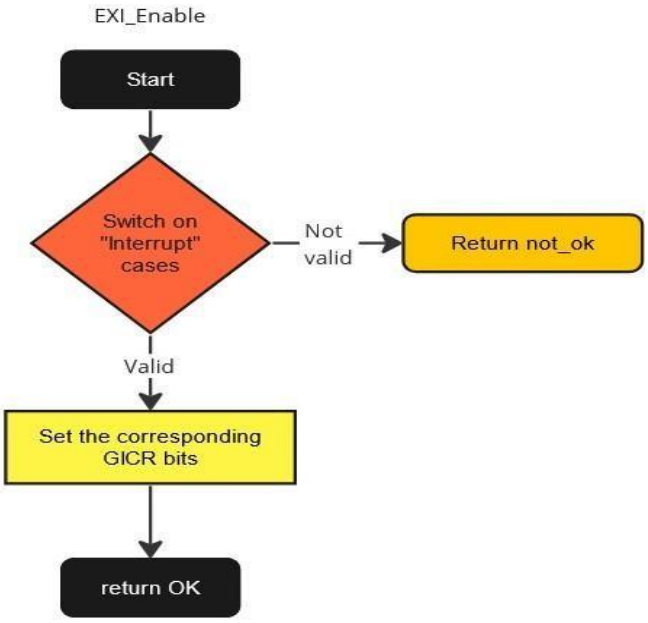


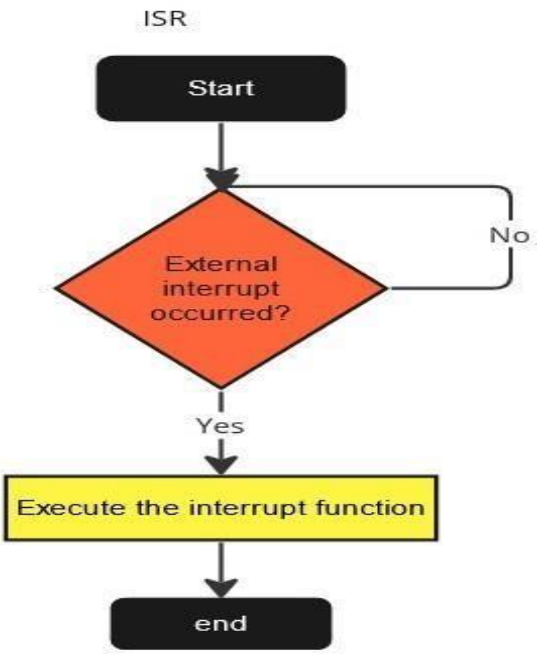




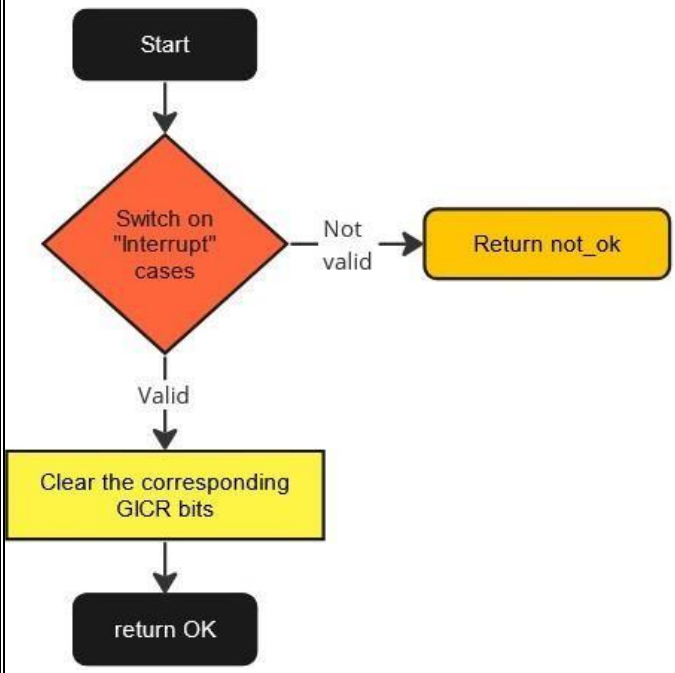


ii. External Interrupt Functions

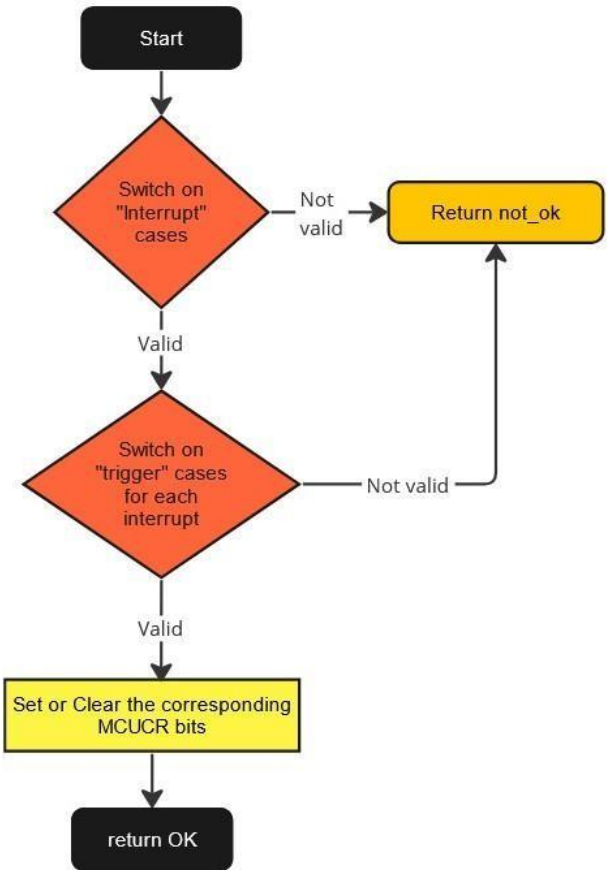




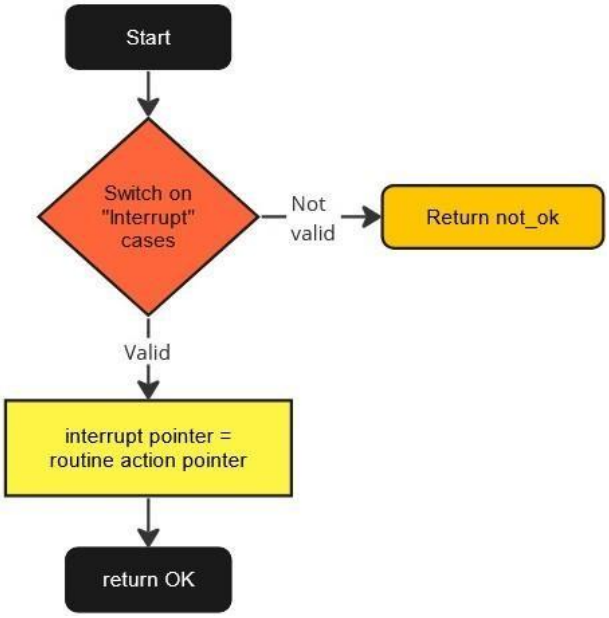
EXI_Disable



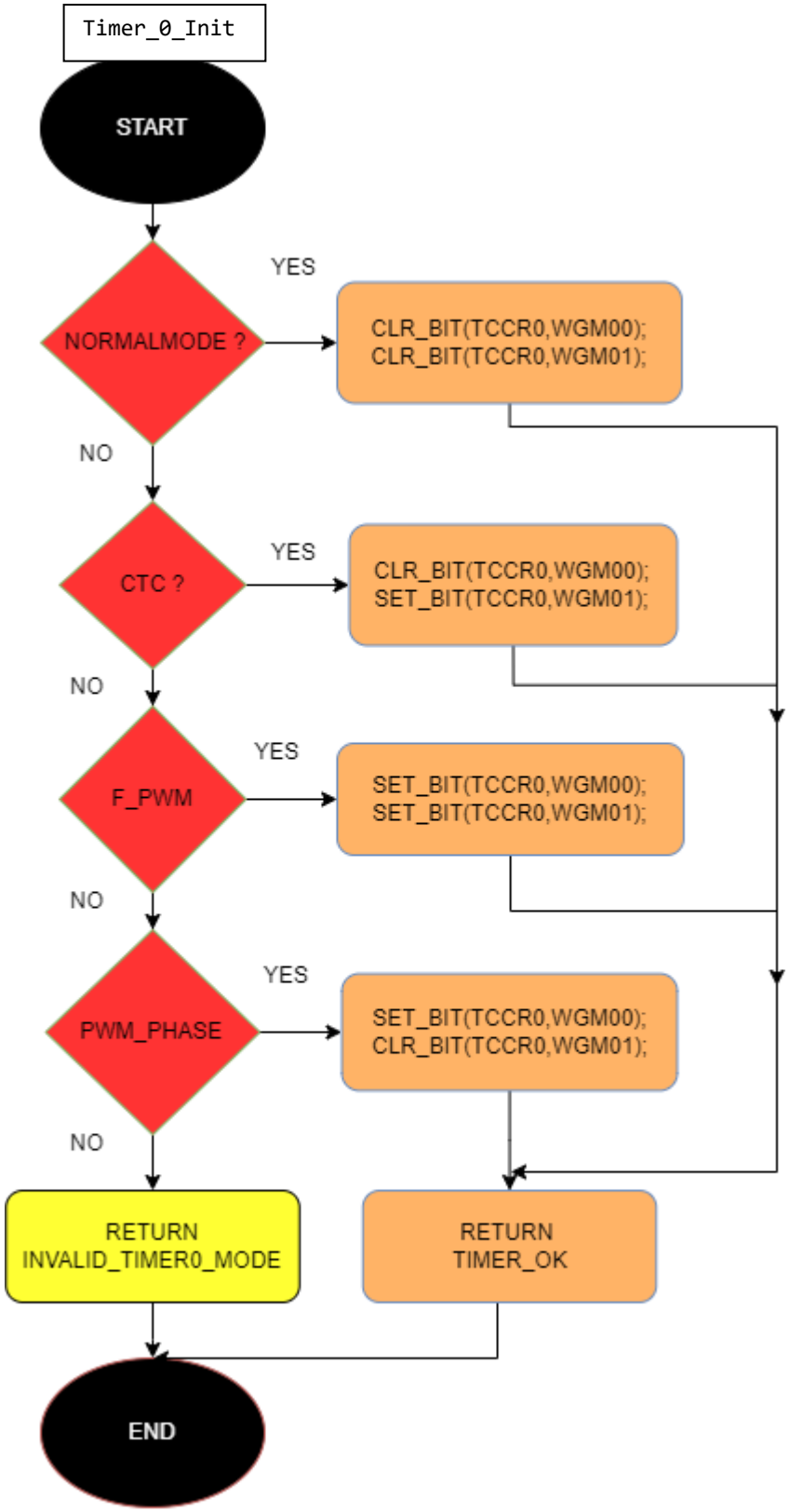
EXI_Trigger



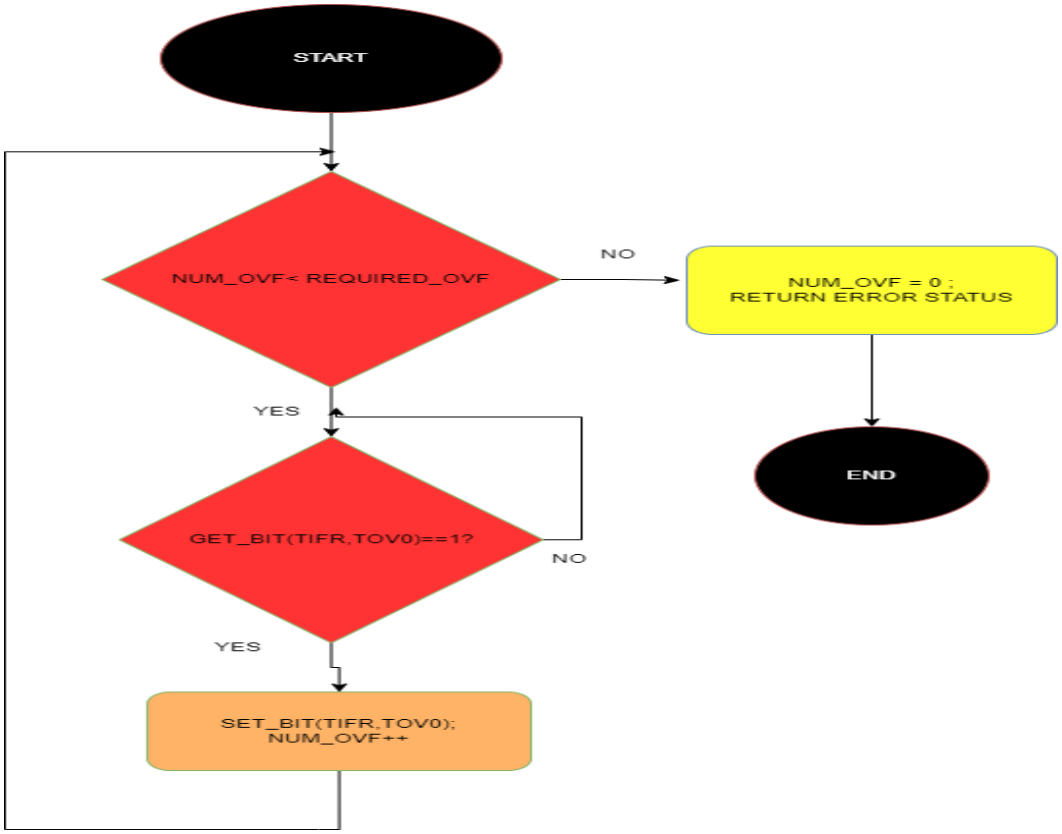
EXI_SetCallBack



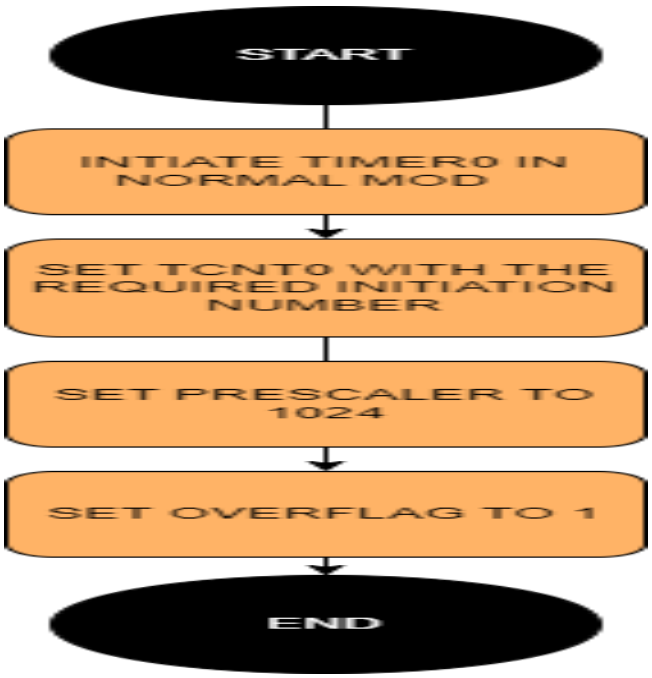
iii. Timers Function



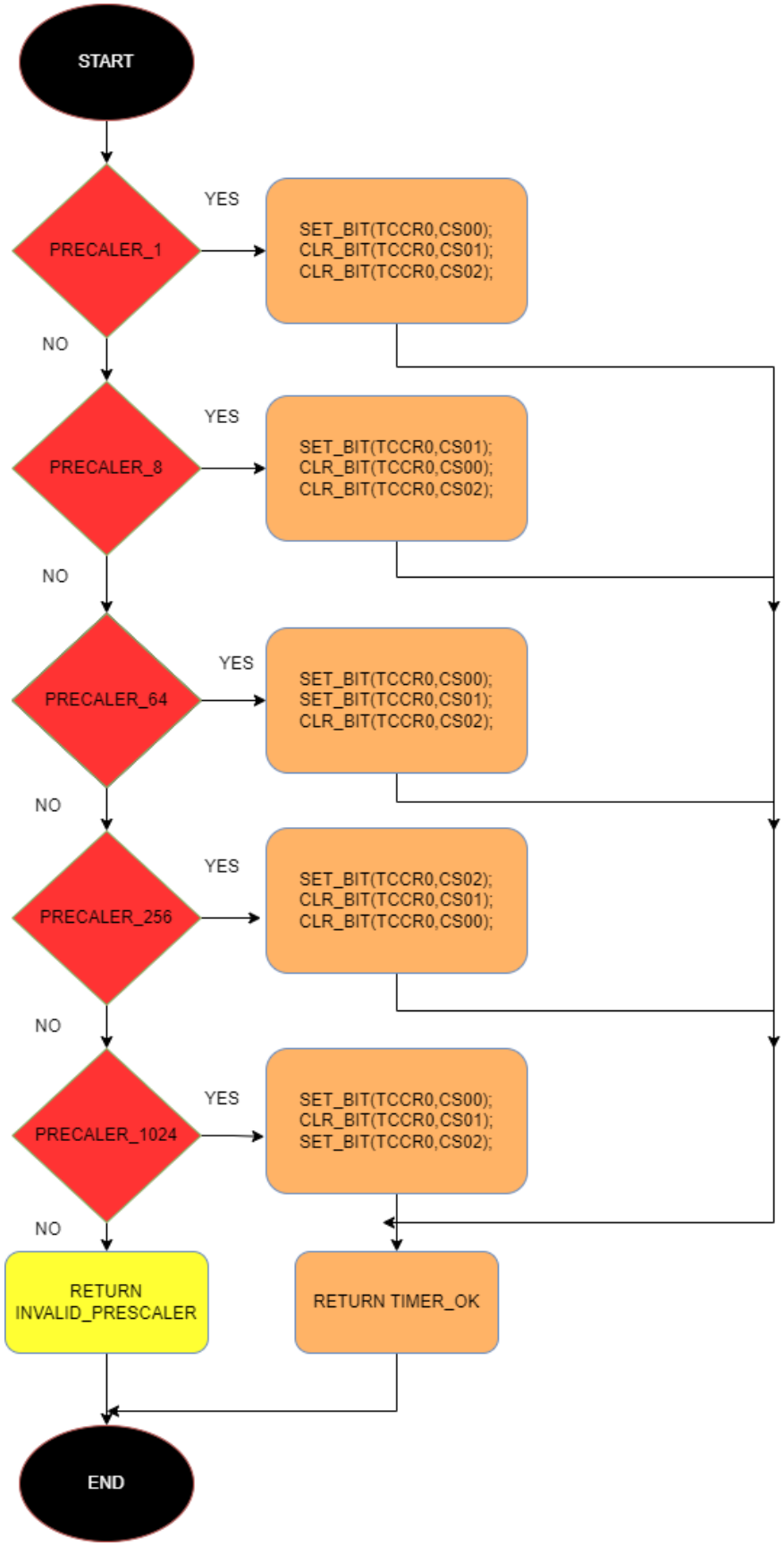
Timer_0_OVFNum



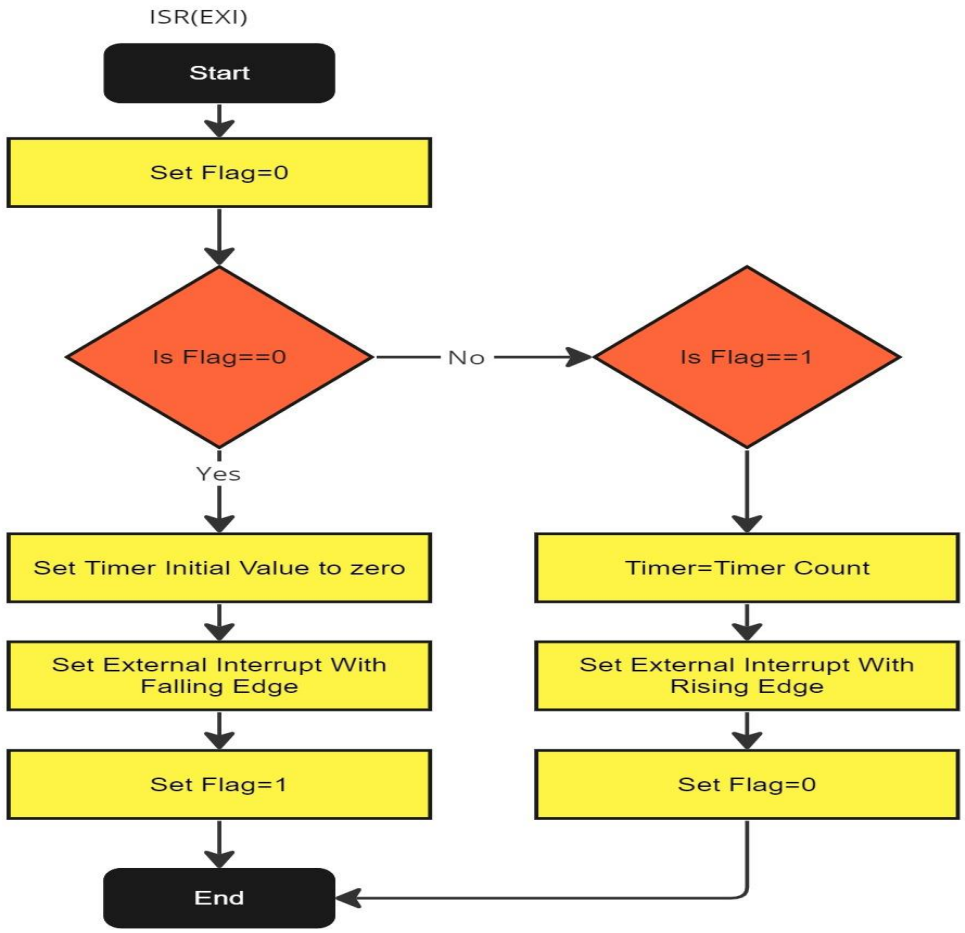
Timer_0_PWM

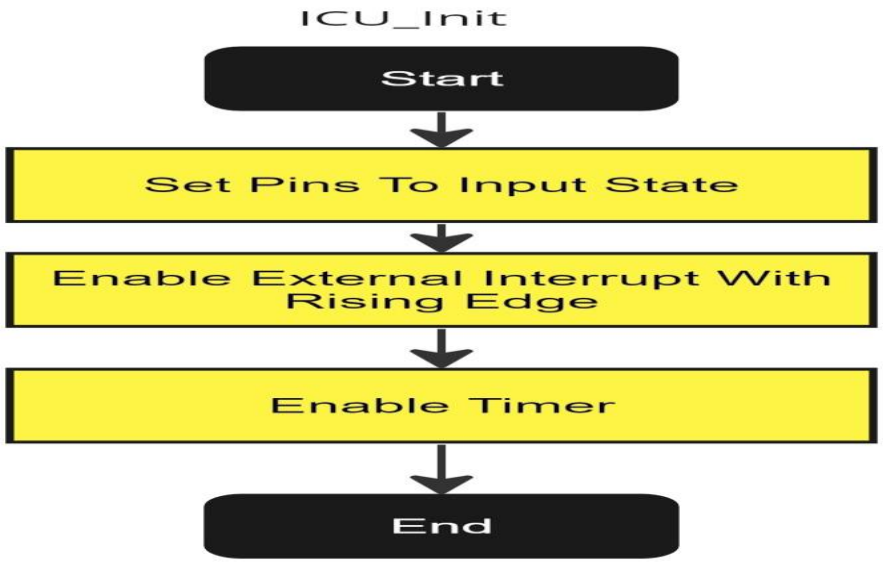


Timer_0_Start



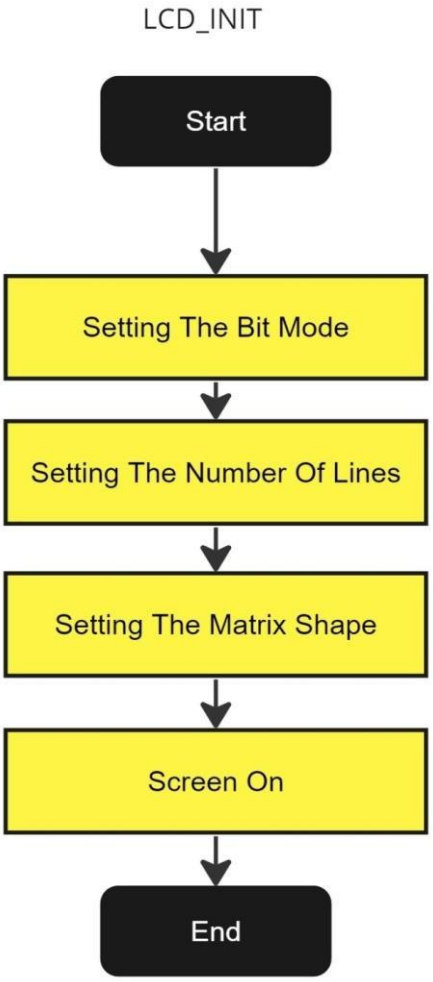
iv. ICU Function:

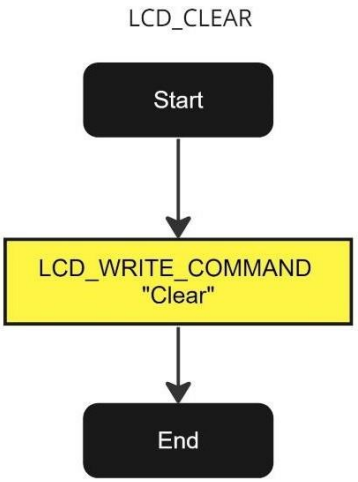




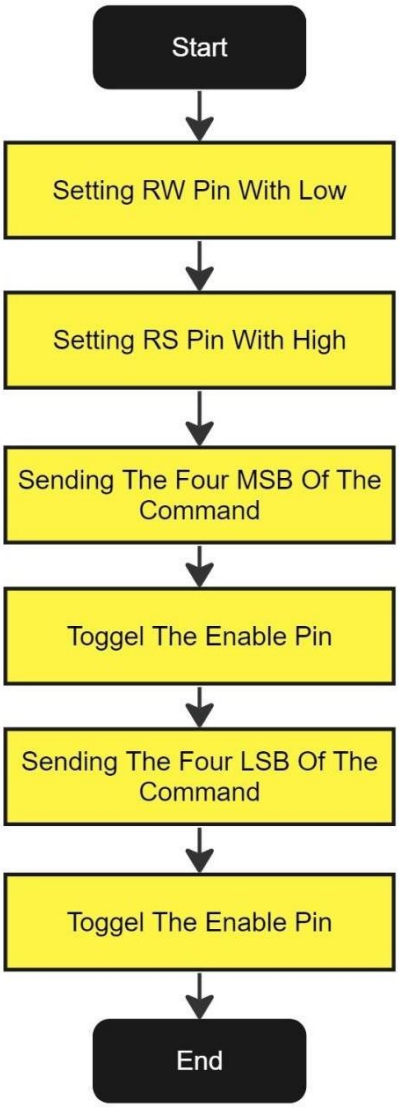
2. HAL Layer:

i. LCD Functions:

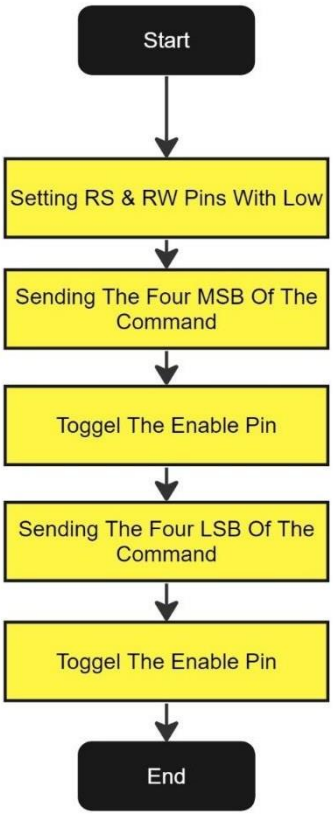




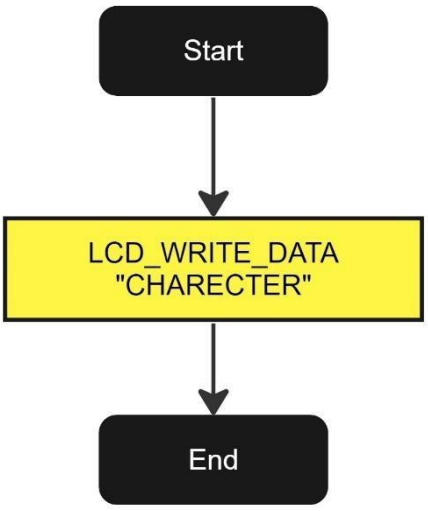
LCD_WRITE_DATA

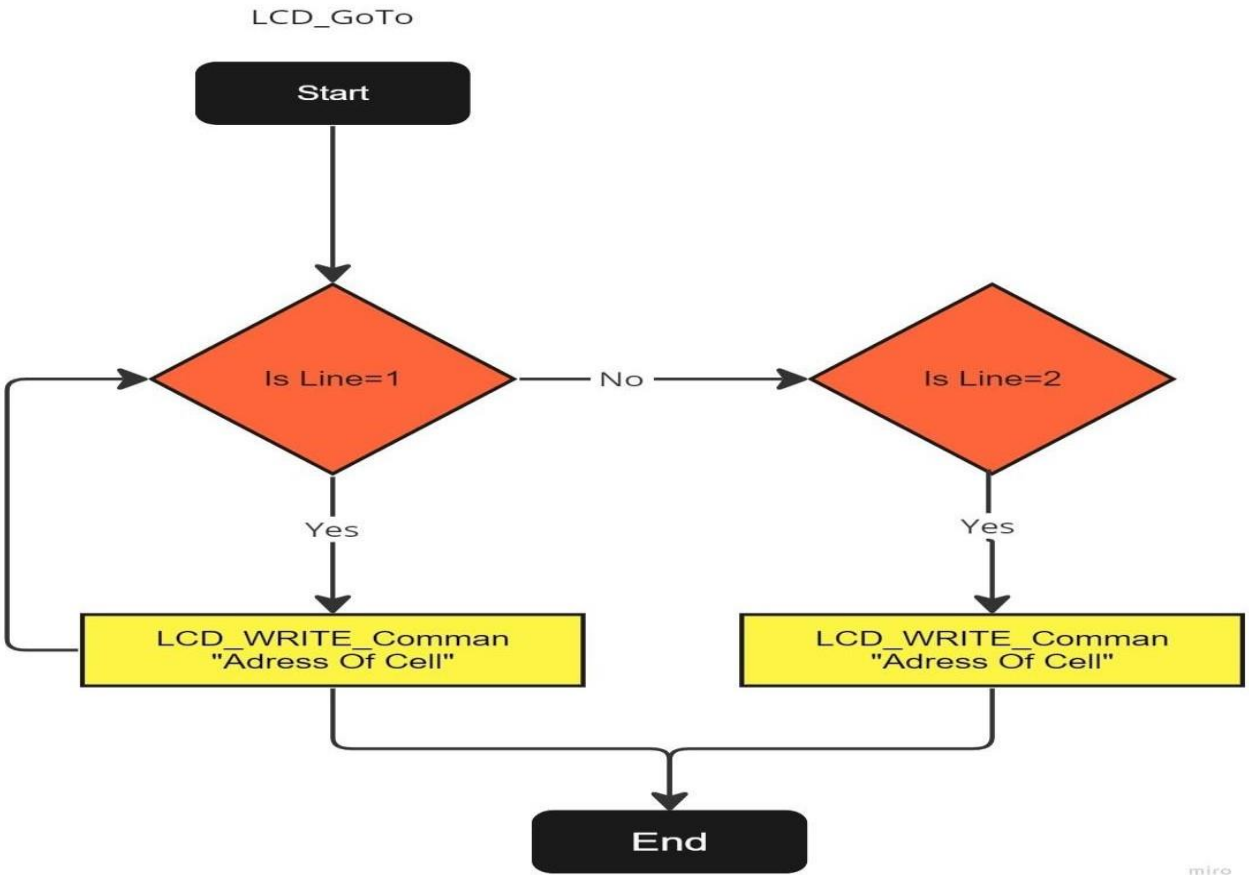


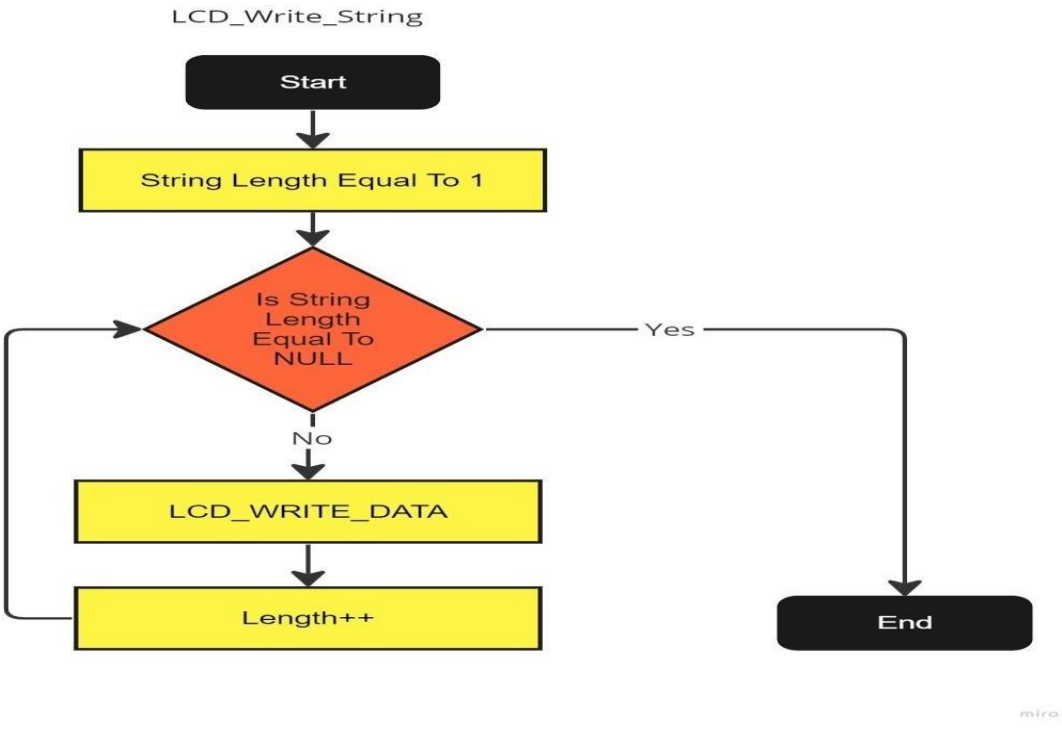
LCD_WRITE_COMMAND



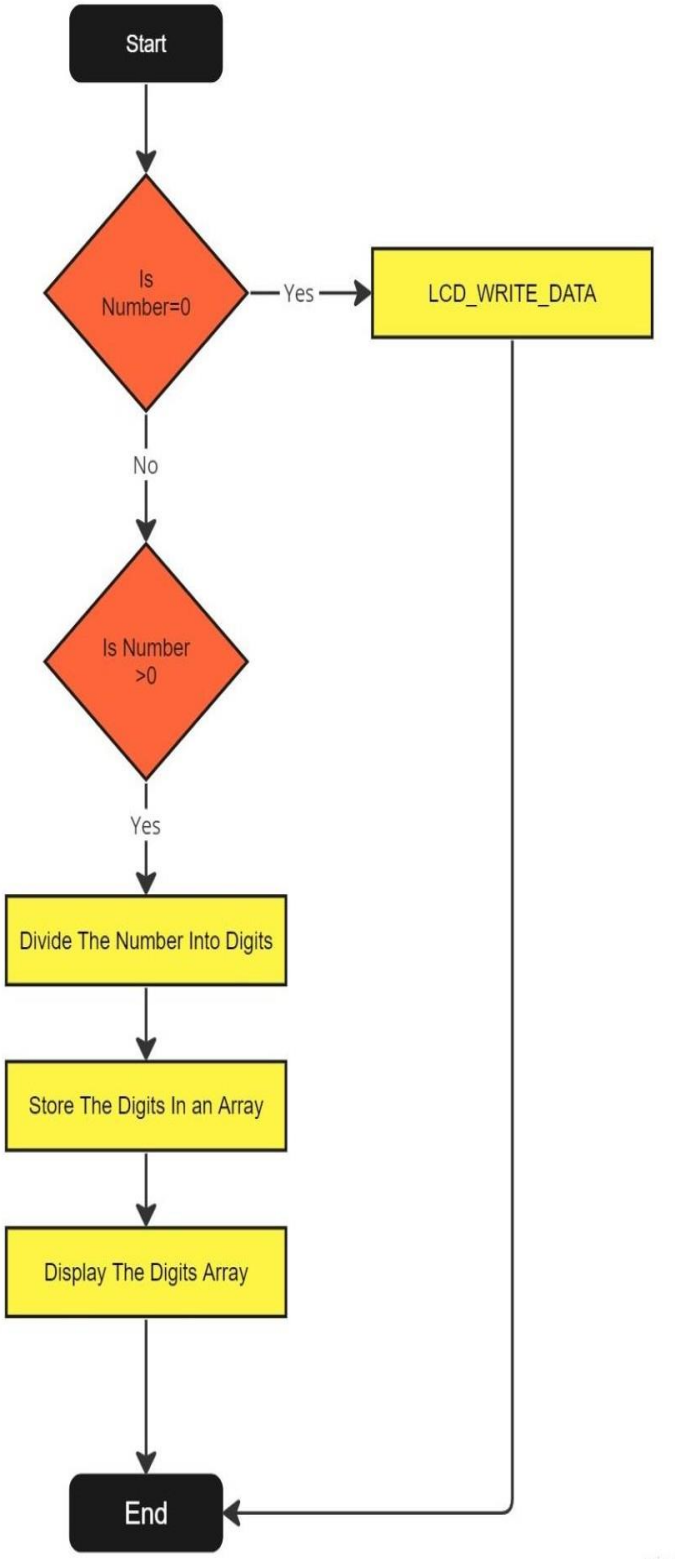
LCD_Write_Charecter

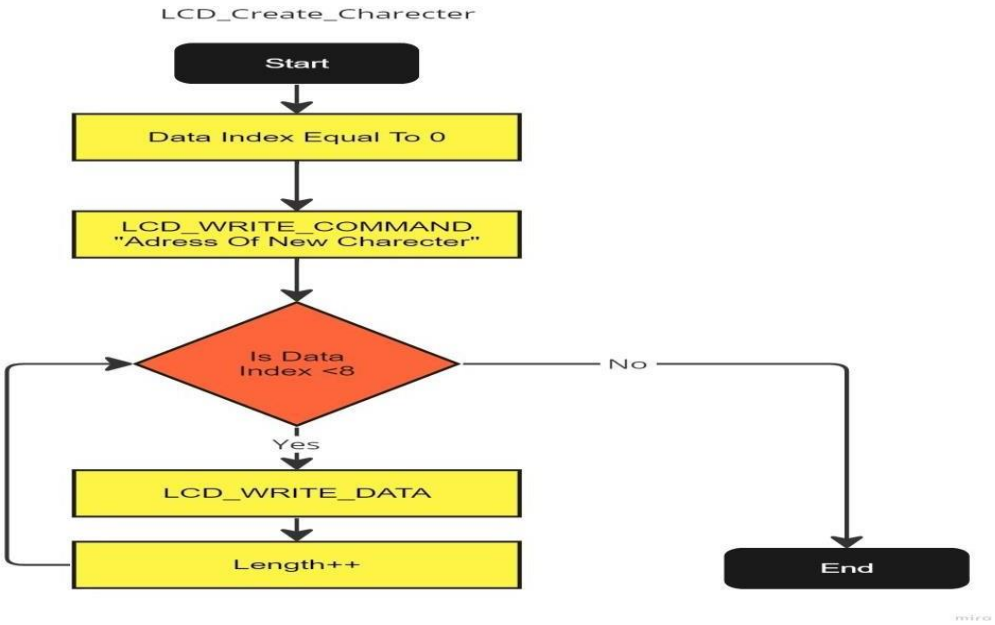



LCD_GoTo
Start
Is Line=1
No
Is Line=2
Yes
Yes
LCD_WRITE_Comman
"Adress Of Cell"
LCD_WRITE_Comman
"Adress Of Cell"
End
miro

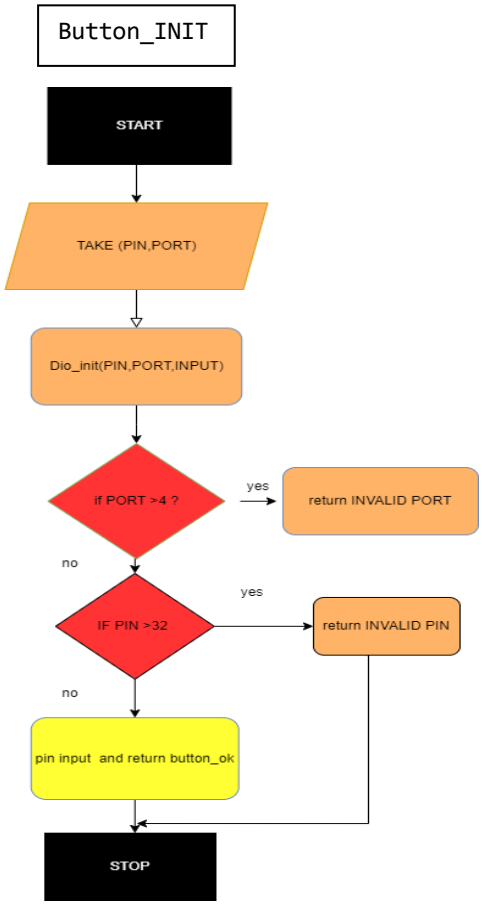

LCD_Write_String
Start
String Length Equal To 1
Is String
Length
Equal To
NULL
Yes
No
LCD_WRITE_DATA
Length++
End
miro

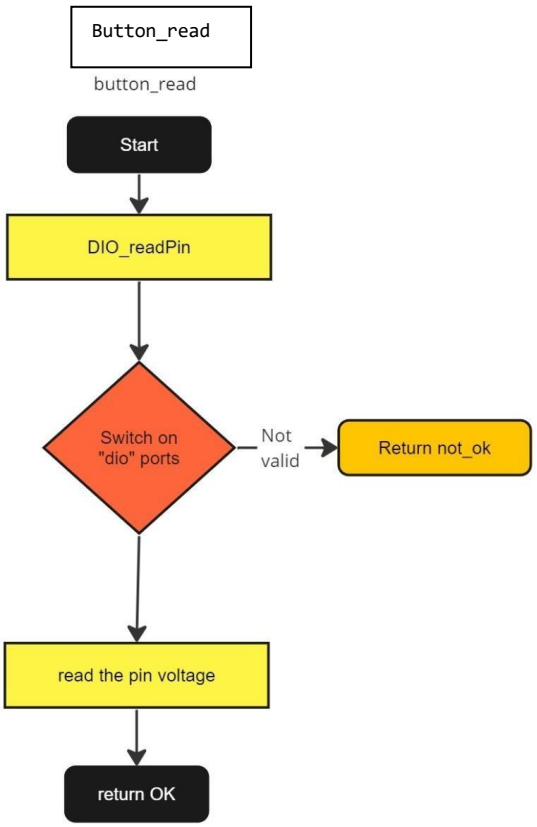
LCD_WRITE_NUMBER



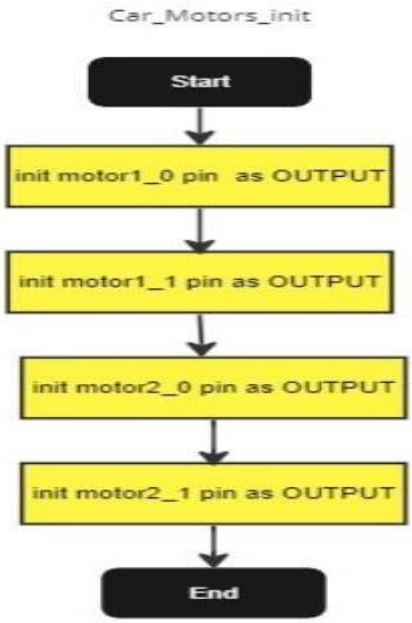


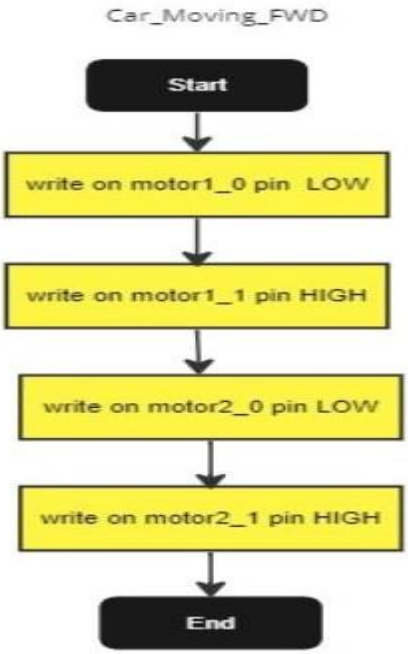
ii. Button Functions:

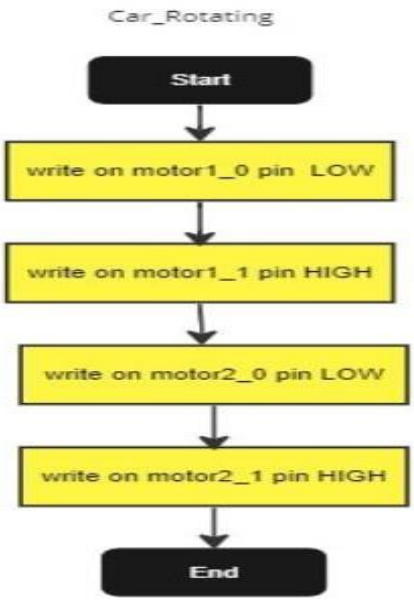


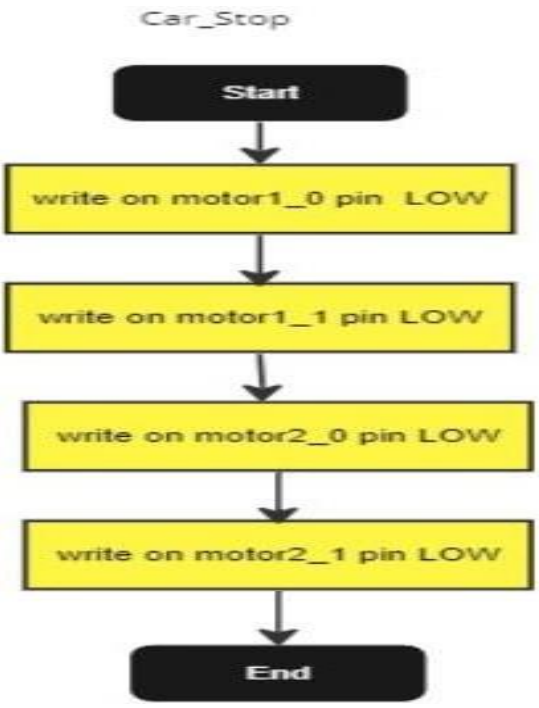


iii. Motor Functions:

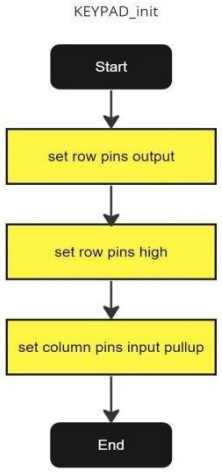


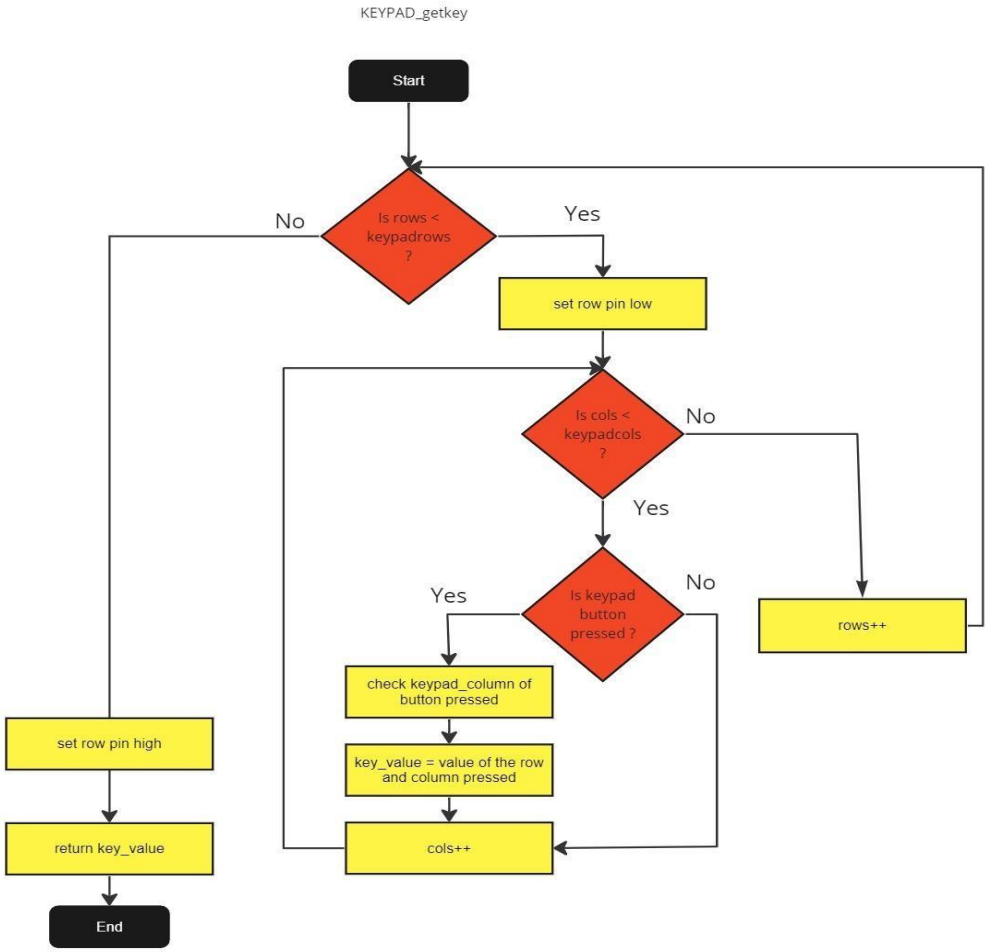



Car_Rotating
Start
write on motor1_0 pin LOW
write on motor1_1 pin HIGH
write on motor2_0 pin LOW
write on motor2_1 pin HIGH
End

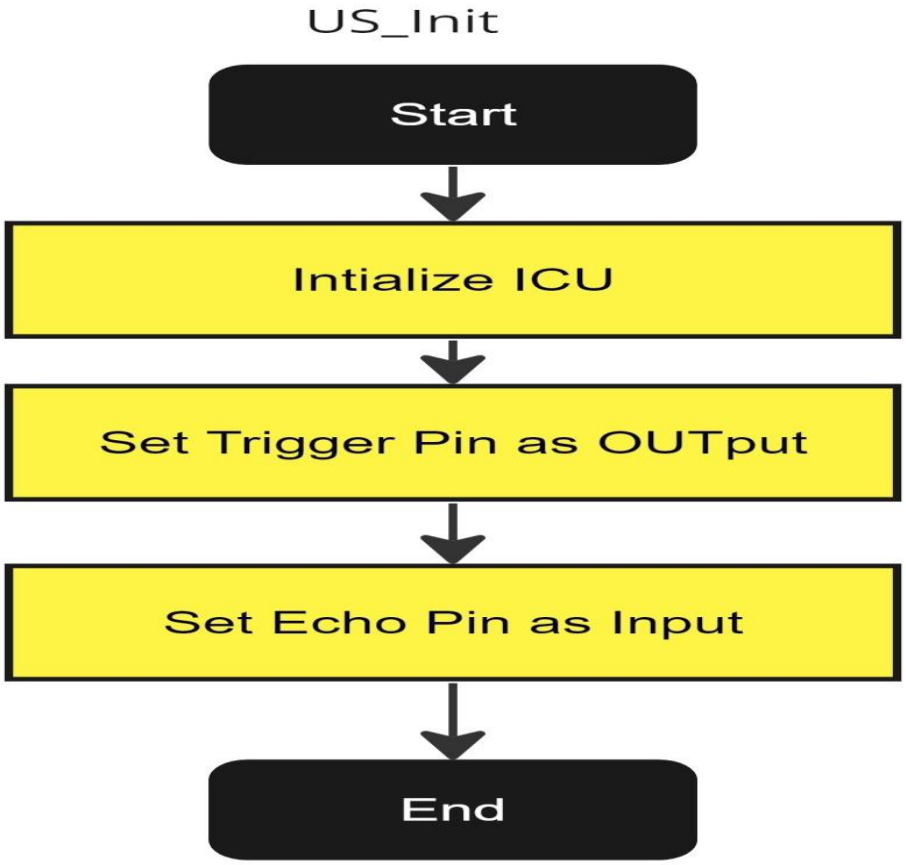

Car_Stop
Start
write on motor1_0 pin LOW
write on motor1_1 pin LOW
write on motor2_0 pin LOW
write on motor2_1 pin LOW
End

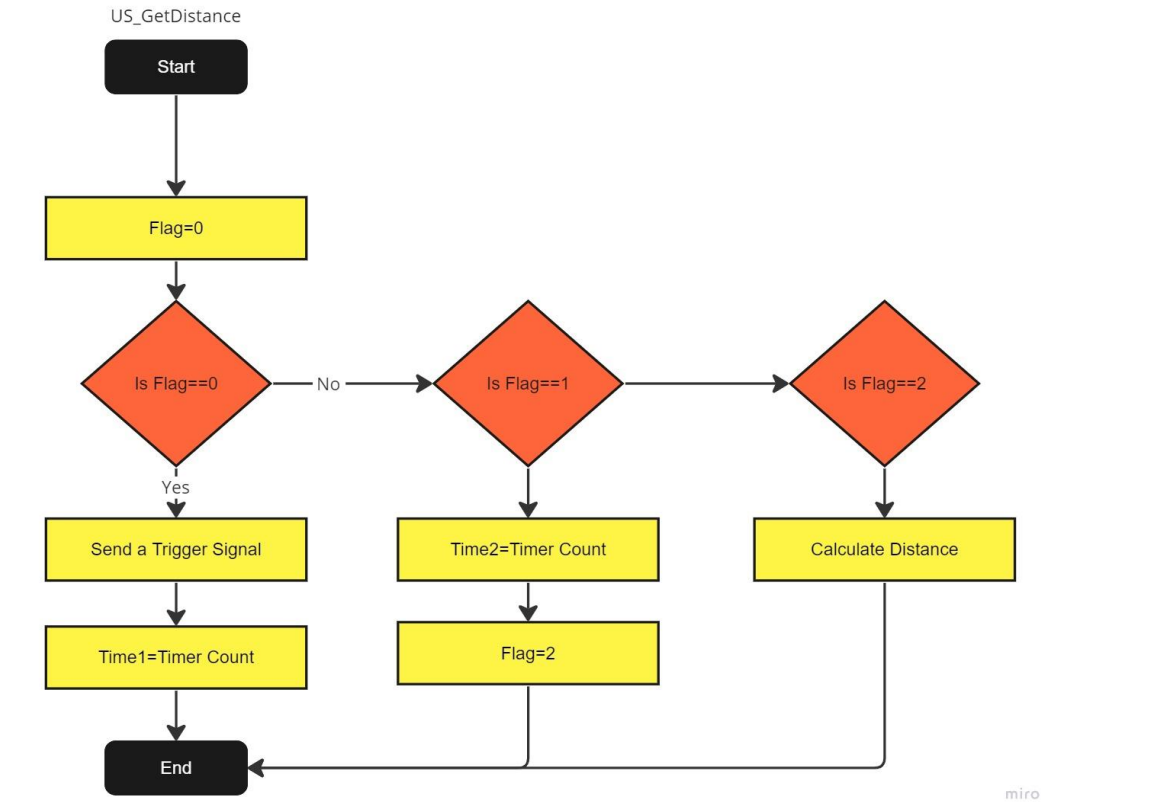
iv. Key Pad Functions:





v. Ultra Sonic Functions:





b. Configurations:-

i. Dio:

```
typedef enum
{
	PINA0=0,
	PINA1,
	PINA2,
	PINA3,
	PINA4,
	PINA5,
	PINA6,
	PINA7,
	PINB0,
	PINB1,
	PINB2,
	PINB3,
	PINB4,
	PINB5,
	PINB6,
	PINB7,
	PINC0,
	PINC1,
	PINC2,
	PINC3,
	PINC4,
	PINC5,
	PINC6,
	PINC7,
	PIND0,
	PIND1,
	PIND2,
	PIND3,
	PIND4,
	PIND5,
	PIND6,
	PIND7,
	TOTAL_PINS=32

}DIO_pinType;
typedef enum
{
	OUTPUT=0,
	INPLUP,
	INFREE
}DIO_PinStatus_Type;
```

```
typedef enum
{
	OUTPUT=0,
	INPLUP,
	INFREE
}DIO_PinStatus_Type;

typedef enum
{
	LOW=0,
	HIGH

}DIO_VoltType;

typedef enum
{
	PA=0,
	PB,
	PC,
	PD
}DIO_PortType;


void DIO_InitPin(DIO_pinType pin,DIO_PinStatus_Type status
void DIO_WritePin(DIO_pinType pin,DIO_VoltType volt);
DIO_VoltType DIO_ReadPin(DIO_pinType pin);
void DIO_TogglePin(DIO_pinType pin);
void DIO_WritePort(DIO_PortType port ,u8 data);
u8 DIO_ReadPort(DIO_PortType port);
extern const DIO_PinStatus_Type PinsStatusArray[TOTAL_PINS
void DIO_Init(void);

#endif /* DIO_H_ */
```

```c
const DIO_PinStatus_Type PinsStatusArray[TOTAL_PINS]=
{
    OUTPUT,   /* PINA0 - ADC0 */
    OUTPUT,   /* PINA1 - ADC1 */
    OUTPUT,   /* PINA2 - ADC2 */
    OUTPUT,   /* PINA3 - ADC3 */
    OUTPUT,   /* PINA4 - ADC4 */
    OUTPUT,   /* PINA5 - ADC5 */
    OUTPUT,   /* PINA6 - ADC6 */
    OUTPUT,   /* PINA7 - ADC7 */
    OUTPUT,   /* PINB0 -      */
    OUTPUT,   /* PINB1 -      */
    OUTPUT,   /* PINB2   INT2 */
    OUTPUT,   /* PINB3 - OC0  */
    OUTPUT,   /* PINB4 - SS   */
    INPLUP,   /* PINB5 - MOSI */
    INPLUP,   /* PINB6 - MISO */
    OUTPUT,   /* PINB7 - SCK  */
    OUTPUT,   /* PINC0 -      */
    OUTPUT,   /* PINC1 -      */
    OUTPUT,   /* PINC2 -      */
    OUTPUT,   /* PINC3 -      */
    OUTPUT,   /* PINC4 -      */
    OUTPUT,   /* PINC5 -      */
    INFREE,   /* PINC6 -      */
    OUTPUT,   /* PINC7 -      */
    OUTPUT,   /* PIND0 - RX   */
    OUTPUT,   /* PIND1 - TX   */
    OUTPUT,   /* PIND2 - INT0 */
    OUTPUT,   /* PIND3 - INT1 */
    OUTPUT,   /* PIND4 - OC1B */
    OUTPUT,   /* PIND5 - OC1A */
    INFREE,   /* PIND6 - ICP1 */
    OUTPUT    /* PIND7 - OC2  */
};
```

ii. EX_Interrupt:

```
typedef enum
{
EX_INT0=0,
EX_INT1,
EX_INT2
} EXInterruptSource_type ;

typedef enum
{
LOW_LEVEL0=0,
ANY_LOGIC_CHANGE,
FALLING_EDGE,
RISING_EDGE
} TriggerEdge_type ;

typedef struct
{
    EXInterruptSource_type EX_INT0;
    TriggerEdge_type       ANY_LOGIC_CHANGE;
    } ST_EXI_Config ;
```

iii. Timers:

```
/* Timer0 */
typedef enum
{
    TIMER0_NORMAL_MODE=0,
    TIMER0_PHASE_CORRECT_MODE,
    TIMER0_CTC_MODE,
    TIMER0_FASTPWM_MODE
}Timer0Mode_type;

typedef enum
{
    TIMER0_STOP=0,
    TIMER0_SCALER_1,
    TIMER0_SCALER_8,
    TIMER0_SCALER_64,
    TIMER0_SCALER_256,
    TIMER0_SCALER_1024,
    TIMER0_EXTERNAL_FALLING,
    TIMER0_EXTERNAL_RISING
}Timer0SCALER_type;

typedef enum
{
    OC0_DISCONNECTED=0,
    OC0_TOGGLE,
    OC0_NON_INVERTING,
    OC0_INVERTING
}OC0Mode_type;

typedef struct
{
    Timer0Mode_type      TIMER0_NORMAL_MODE;
    Timer0SCALER_type    TIMER0_SCALER_8;
    OC0Mode_type         OC0_DISCONNECTED

    }ST_Timer_Config;
```

iv. LCD:

```
/************************ config*************************************************/
#define _4_bit_mode     0
#define _8_bit_mode     1
#define LCD_Mode       _4_bit_mode

#define  LCD_PORT      PC

#define   D4           pinc0
#define   D5           pinc1
#define   D6           pinc2
#define   D7           pinc3

#define   RS           pinc6
#define   RW           pinc5
#define   EN           pinc4

/*******************************************************************************/

/*************************************************************************************************************
                                                PROTOTYPES
*************************************************************************************************************/

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);
void LCD_Create_Charecter(uint8_t*a_Pattern,uint8_t a_Adress);
/*************************************************************************************************************/
#endif /* LCD_H_ */
```

```
#if (LCD_Mode==_8_bit_mode)

/*************************************************************************************************
                                    8_bit_mode
*************************************************************************************************/
/**DESCRIPTION:-
   This Function Send a Command To The LCD MicroController To Be Executed
   **/
void LCD_WRITE_COMMAND(uint8_t a_COMMAND)
{
    DIO_WRITEPIN(RS,LOW);
    DIO_WRITEPIN(RW,LOW);
    DIO_WritePort(LCD_PORT,a_COMMAND);
    DIO_WRITEPIN(EN,HIGH);
    _delay_ms(1);
    DIO_WRITEPIN(EN,LOW);
    _delay_ms(1);

}

/*************************************************************************************************************/
/**DESCRIPTION:-
   This Function Send Data To The LCD MicroController To Be diplayed
   **/
void LCD_WRITE_DATA(uint8_t a_DATA)
{
    DIO_WRITEPIN(RS,HIGH);
    DIO_WRITEPIN(RW,LOW);
    DIO_WritePort(LCD_PORT,a_DATA);
    DIO_WRITEPIN(EN,HIGH);
    _delay_ms(1);
    DIO_WRITEPIN(EN,LOW);
    _delay_ms(1);

}

/*************************************************************************************************************/
/**DESCRIPTION:-
   This Function Initializes The LCD With the required Configuration
   **/
void LCD_INIT(void)
{
    _delay_ms(50);

    LCD_WRITE_COMMAND(0X38);                    /* THIS IS FOR UNCTION SET IN THE DATA SHEET  8BIT MODE, 2LINE, 5*7*/
    _delay_ms(1);
    LCD_WRITE_COMMAND(0x0c);                    /*SCREEN ON, CURSOR OFF*/
    _delay_ms(1);
    LCD_WRITE_COMMAND(0X01);                     /*CLEAR SCREEN*/
    _delay_ms(2);
    LCD_WRITE_COMMAND(0X06);
    _delay_ms(1);
}
```

```c
#elif (LCD_Mode==_4_bit_mode)

/*************************************************************************************************
                                    4_bit_mode
*************************************************************************************************/

void LCD_WRITE_COMMAND(uint8_t a_COMMAND)
{
    DIO_WRITEPIN(RS,LOW);
    DIO_WRITEPIN(RW,LOW);
    /**for the 4 most significant bits**/
    DIO_WRITEPIN(D4,read_bit(a_COMMAND,4));
    DIO_WRITEPIN(D5,read_bit(a_COMMAND,5));
    DIO_WRITEPIN(D6,read_bit(a_COMMAND,6));
    DIO_WRITEPIN(D7,read_bit(a_COMMAND,7));
    DIO_WRITEPIN(EN,HIGH);
    _delay_ms(1);
    DIO_WRITEPIN(EN,LOW);
    _delay_ms(1);
    /**for the 4 least significant bits**/
    DIO_WRITEPIN(D4,read_bit(a_COMMAND,0));
    DIO_WRITEPIN(D5,read_bit(a_COMMAND,1));
    DIO_WRITEPIN(D6,read_bit(a_COMMAND,2));
    DIO_WRITEPIN(D7,read_bit(a_COMMAND,3));
    DIO_WRITEPIN(EN,HIGH);
    _delay_ms(1);
    DIO_WRITEPIN(EN,LOW);
    _delay_ms(1);
}

/*************************************************************************************************************************/

void LCD_WRITE_DATA(uint8_t a_DATA)
{
    DIO_WRITEPIN(RS,HIGH);
    DIO_WRITEPIN(RW,LOW);
    /**for the 4 most significant bits**/
    DIO_WRITEPIN(D4,read_bit(a_DATA,4));
    DIO_WRITEPIN(D5,read_bit(a_DATA,5));
    DIO_WRITEPIN(D6,read_bit(a_DATA,6));
    DIO_WRITEPIN(D7,read_bit(a_DATA,7));
    DIO_WRITEPIN(EN,HIGH);
    _delay_ms(1);
    DIO_WRITEPIN(EN,LOW);
    _delay_ms(1);
    /**for the 4 least significant bits**/
    DIO_WRITEPIN(D4,read_bit(a_DATA,0));
    DIO_WRITEPIN(D5,read_bit(a_DATA,1));
    DIO_WRITEPIN(D6,read_bit(a_DATA,2));
    DIO_WRITEPIN(D7,read_bit(a_DATA,3));
    DIO_WRITEPIN(EN,HIGH);
    _delay_ms(1);
    DIO_WRITEPIN(EN,LOW);
    _delay_ms(1);
}
/*************************************************************************************************************************/
```