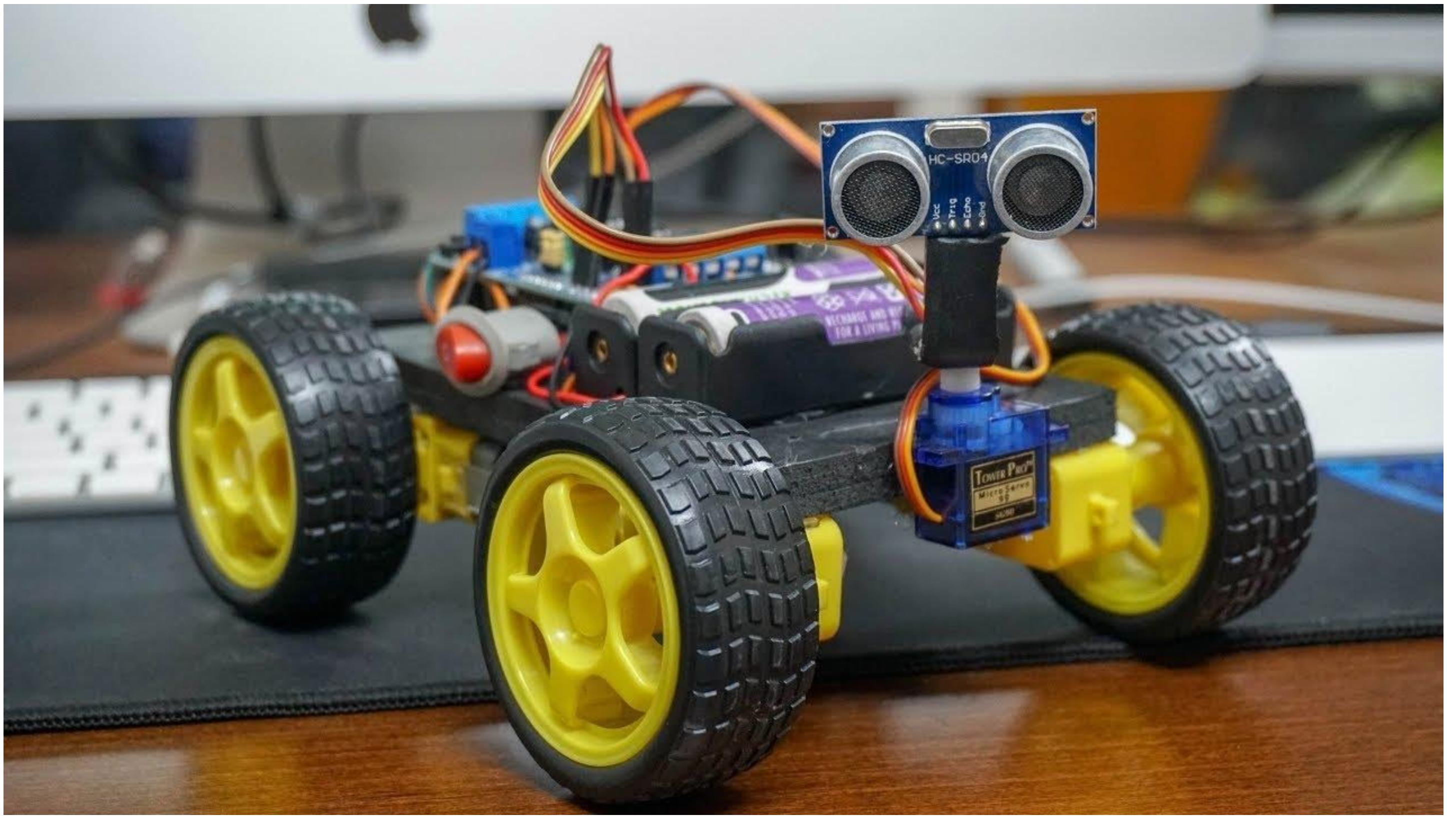
Obstacle Avoidance Robot V1.0 Design



Team 2

1- Ahmed Atef

2- Amr Gamal El-Abd

3- Anas Mahmoud

4- Khaled Mustafa

Table of Contents

1-Description

1.1 Hardware components

1. Car Components:

    1. ATmega32 microcontroller

    2. Four motors (M1, M2, M3, M4)

    3. One button to change default direction of rotation (PBUTTON0)

    4. Keypad button 1 to start

    5. Keypad button 2 to stop

    6. One Ultrasonic sensor connected as follows 1. Vcc to 5V in the Board 2. GND to the ground In the Board 3. Trig to PB3 (Port B, Pin 3) 4. Echo to PB2 (Port B, Pin 2)

7. LCD

1.2 System Requirements

1. The car starts initially from 0 speed

2. The default rotation direction is to the right

3. Press (Keypad Btn 1), (Keypad Btn 2) to start or stop the robot respectively

4. After Pressing Start:

    1. The LCD will display a centered message in line 1 “Set Def. Rot.”

    2. The LCD will display the selected option in line 2 “Right”

    3. The robot will wait for 5 seconds to choose between Right and Left

        1. When PBUTTON0 is pressed once, the default rotation will be Left and the LCD line 2 will be updated

        2. When PBUTTON0 is pressed again, the default rotation will be Right and the LCD line 2 will be updated

        3. For each press the default rotation will changed and the LCD line 2 is updated

    4. After the 5 seconds the default value of rotation is set

4. The robot will move after 2 seconds from setting the default direction of rotation.

5. For No obstacles or object is far than 70 centimeters:

    1. The robot will move forward with 30% speed for 5 seconds

    2. After 5 seconds it will move with 50% speed as long as there was no object or objects are located at more than 70 centimeters distance

3. The LCD will display the speed and moving direction in line 1: “Speed:00% Dir: F/B/R/S”, F: forward, B: Backwards, R: Rotating, and S: Stopped

4. The LCD will display Object distance in line 2 “Dist.: 000 Cm”

6. For Obstacles located between 30 and 70 centimeters

   1. The robot will decrease its speed to 30%

   2. LCD data is updated

7. For Obstacles located between 20 and 30 centimeters

   1. The robot will stop and rotates 90 degrees to right/left according to the chosen configuration

   2. The LCD data is updated

8. For Obstacles located less than 20 centimeters

   1. The robot will stop, move backwards with 30% speed until distance is greater than 20 and less than 30

   2. The LCD data is updated

   3. Then preform point 8

9. Obstacles surrounding the robot (Bonus)

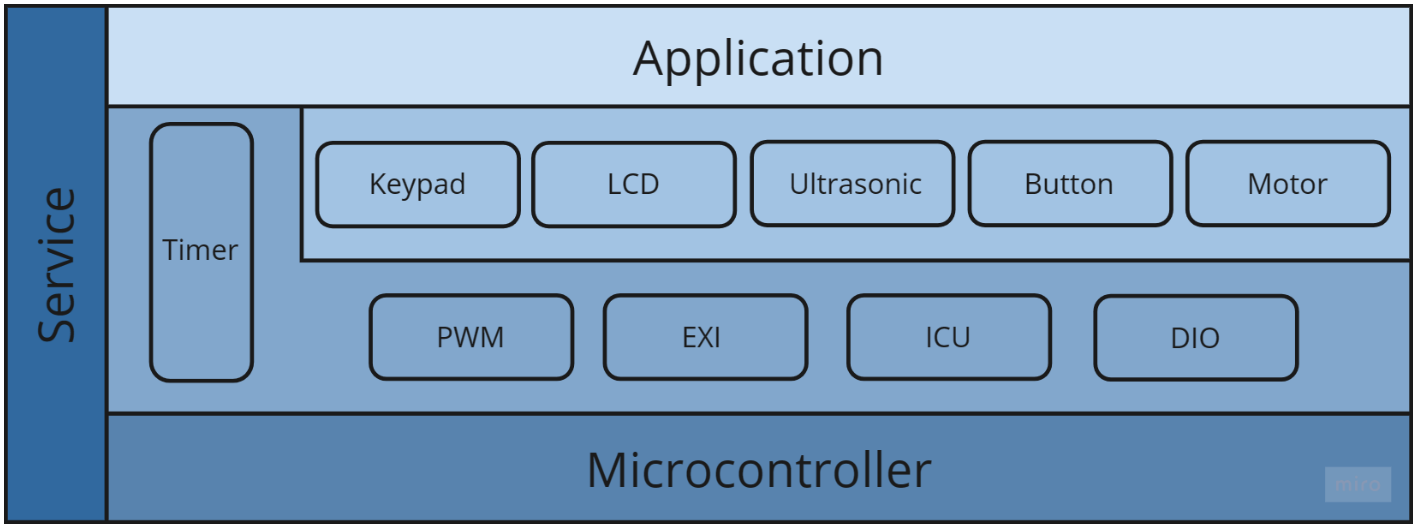
   1. If the robot rotated for 360 degrees without finding any distance greater than 20 it will stop

   2. LCD data will be updated.

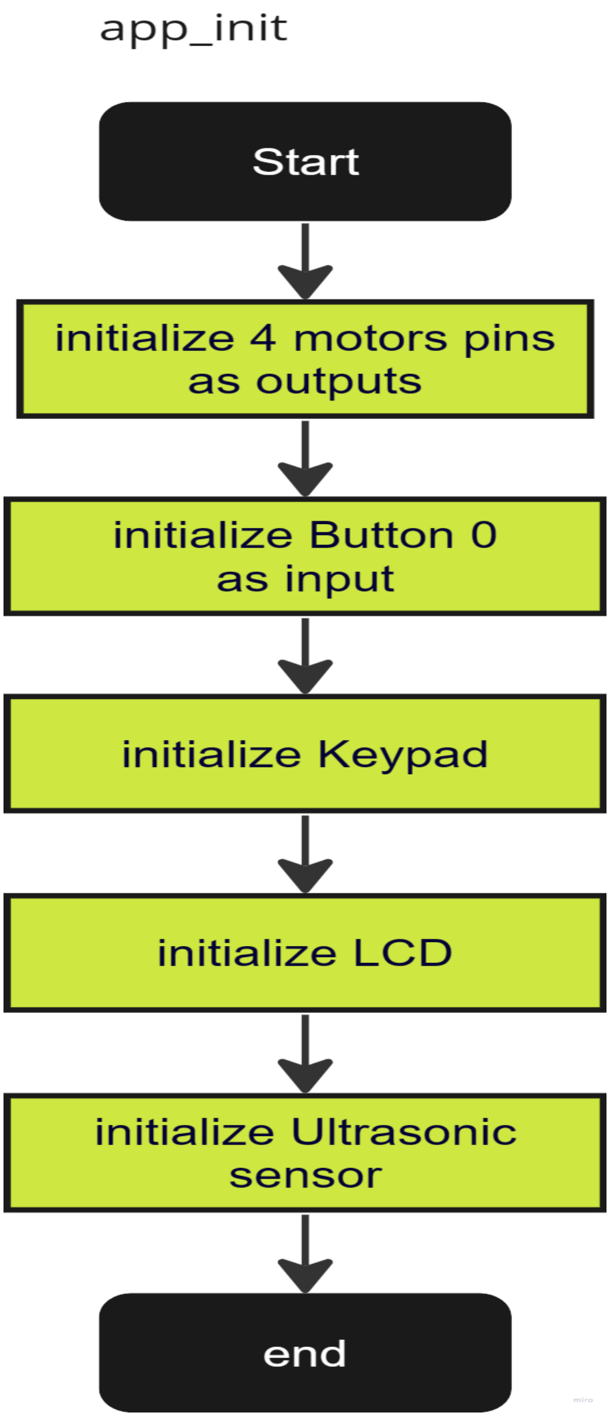
   3. The robot will frequently (each 3 seconds) check if any of the obstacles was removed or not and move in the direction of the furthest object

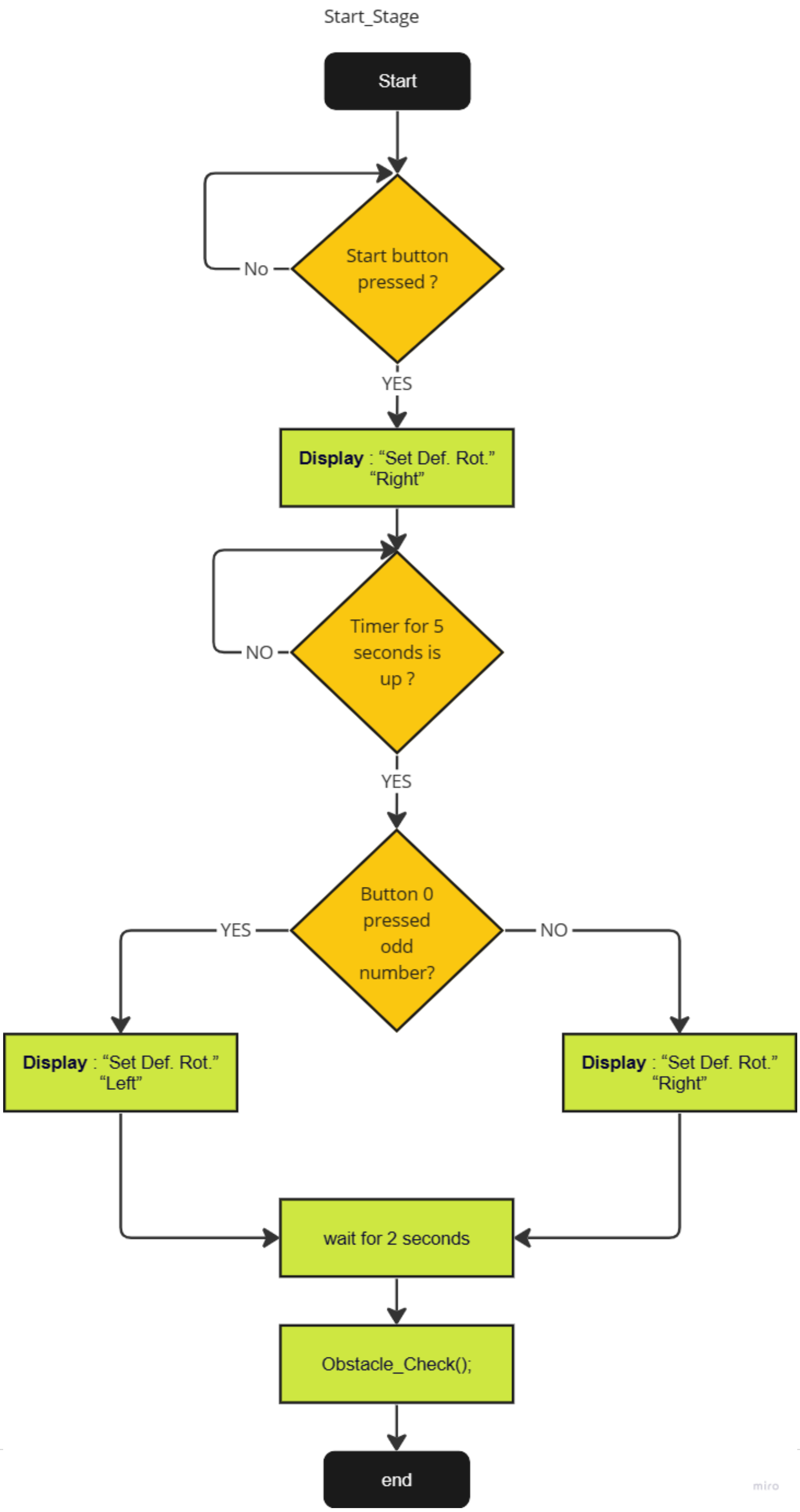
2-High Level Design

2.1 Layered Architecture

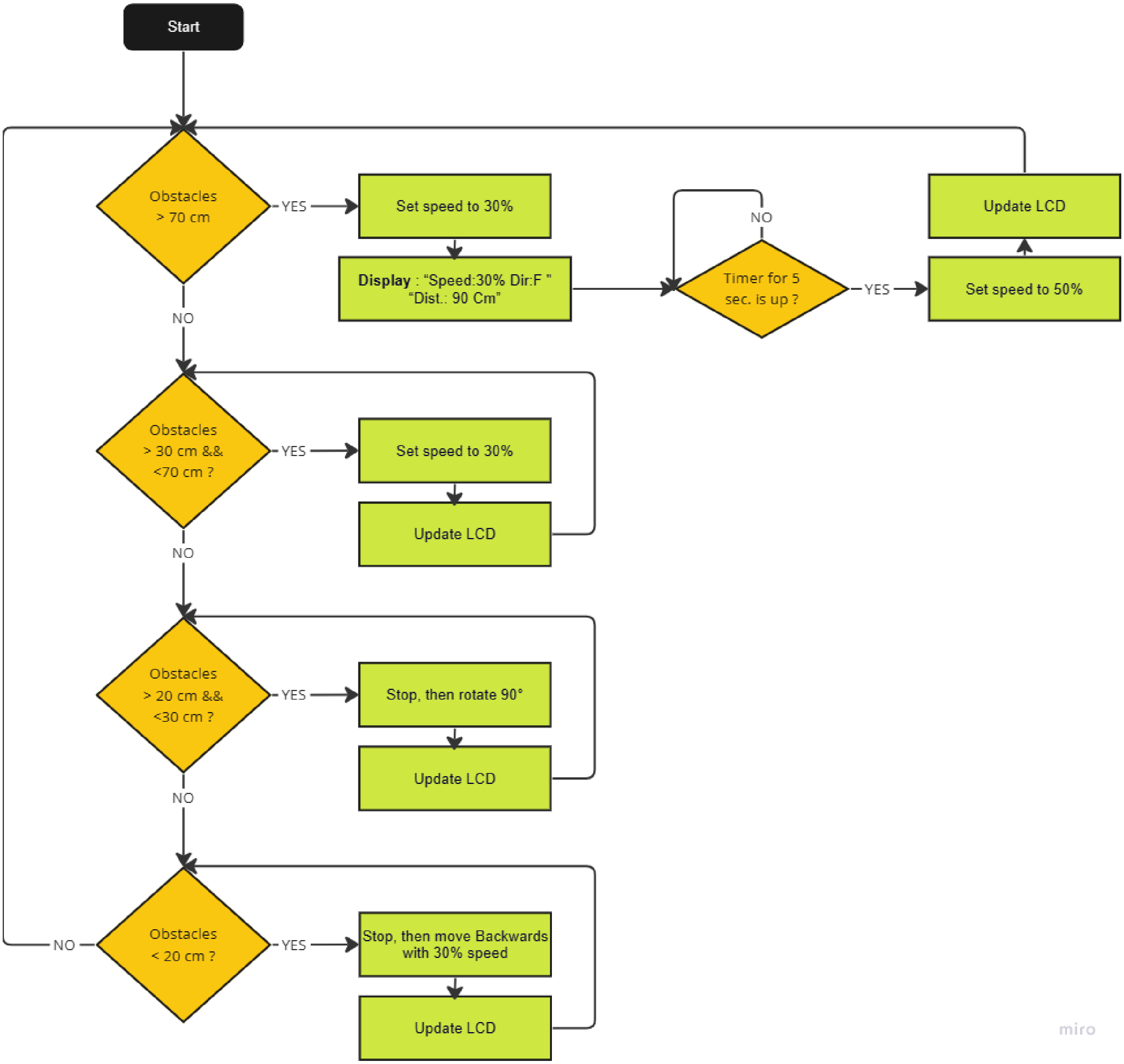


2.2 System Flow Chart


app_init
Start
initialize 4 motors pins as outputs
initialize Button 0 as input
initialize Keypad
initialize LCD
initialize Ultrasonic sensor
end


Start_Stage
Start
Start button pressed ?
No
YES
Display : "Set Def. Rot." "Right"
Timer for 5 seconds is up ?
NO
YES
Button 0 pressed odd number?
YES
NO
Display : "Set Def. Rot." "Left"
Display : "Set Def. Rot." "Right"
wait for 2 seconds
Obstacle_Check();
end
miro

Obstacle_Check();



2.3 Drivers Descriptions

2.3.1 DIO Driver

Configuration: Consist of 6 API’s

Location: MCAL

Function: used to set pin direction (input or output), pin value (high or low) or read a value from a pin or toggle a pin

2.3.2 Timer Driver

Configuration: Consist of 6 API’s

Location: MCAL

Function: used to set a time delay

2.3.3 PWM Driver

Configuration: Consist of 5 API’s

Location: MCAL

Function: used to control motor speed

2.3.4 EXI Driver

Configuration: Consist of 5 API’s

Location: MCAL

Function: used to handle external events that happen during the execution

2.3.5 ICU Driver

Configuration: Consist of 5 API’s

Location: MCAL

Function: The Input Capture Unit Module is used to measure time between to events

2.3.6 Keypad Driver

Configuration: Consist of 2 API’s

Location: HAL

Function: used to initialize the keypad, get pressed key

2.3.7 Button Driver

Configuration: Consist of 1 API’s

Location: HAL

Function: used to check the button status pressed or not

2.3.8 LCD Driver

Configuration: Consist of 12 API’s

Location: HAL

Function: used to initialize the LCD, send command to LCD & display character or string to LCD & jump to specific position on LCD & to clear the LCD & to wright integer or float number on the LCD

2.3.9 Ultrasonic Driver

Configuration: Consist of 4 API’s

Location: HAL

Function: used to detect the distance between car & obstacle

2.3.10 Motor Driver

Configuration: Consist of 6 API’s

Location: HAL

Function: used to control car moving

2.3.11 Application Driver

Configuration: Consist of 2 API’s

Location: App

Function: combine between the drivers API’s to meet the requirement

2.4 Modules API's

2.4.1 DIO Module

1- void DIO_Init_All(void);

2- en_dioError_t DIO_initpin (DIO_Pin_type pin,DIO_PinStatus_type status);

3- en_dioError_t DIO_writepin (DIO_Pin_type pin,DIO_PinVoltage_type volt);

4- en_dioError_t DIO_readpin (DIO_Pin_type pin,DIO_PinVoltage_type *volt);

5- en_dioError_t DIO_togglepin(DIO_Pin_type pin);

6- en_dioError_t DIO_WritePort (DIO_Port_type port,u8 value);

2.4.2 Timer Module

1- EN_timerError_t TIMER_2_init(Timer2Mode_type a_mode);

2- EN_timerError_t TIMER_2_start(Timer2Scaler_type a_prescaler);

3- void TIMER_2_stop(void);

4- EN_timerError_t TIMER_2_setIntialValue(u8 a_value);

5- EN_timerError_t TIMER_2_OvfNum(double overflow);

6- void TIMER_2_DELAY_MS(double _delay);

2.4.3 PWM Module

1- void pwm_init();

2- void waveGen();

3- void PWM_set_duty(u8 u8_DutyCycle,u32 u32_Freq);

4- void waveGen_hf();

5- void PWM_set_duty_hf(u8 u8_DutyCycle,u32 u32_Freq);

2.4.4 EXI Module

1- void EXI_Init(void);

2- void EXI_Enable(ExInterruptSource_type Interrupt);

3- void EXI_Disable(ExInterruptSource_type Interrupt);

4- void EXI_TriggerEdge(ExInterruptSource_type Interrupt,TriggerEdge_type Edge);

5- void EXI_SetCallBack(ExInterruptSource_type Interrupt,void(*LocalPtr)(void));

2.4.5 ICU Module

1- void PWM_Measure(u32* Pfreq,u8* Pduty);

2- static void Func_ICU(void);

3- void PWM_Measure_exi(u32* Pfreq,u8* Pduty);

4- static void Func_ICU_exi(void);

5- void ICU_SW(u32* Pfreq,u8* Pduty);

2.4.6 Keypad Module

1- KEYPAD_initError KEYPAD_init(void) ;

2- KEYPAD_readError KEYPAD_getpressedkey(u8 *value) ;

2.4.7 Button Module

1- Button_State Is_pressed(u8 BUTTON_PIN , u8 *value);

2.4.8 LCD Module

1- void LCD_Init(void);

2- void LCD_WriteChar(u8 ch);

3- void LCD_WriteString(u8*str);

4- void LCD_WriteNumber(s32 num);

5- void LCD_WriteBinary(u8 num);

6- void LCD_WriteHex(u8 num);

7- void LCD_SetCursor(u8 line,u8 cell);

8- void LCD_Clear(void);

9- void LCD_ClearLoc(u8 line ,u8 cell,u8 num);

10- void LCD_WriteNumber_3D(u16 num);

11- void LCD_CustomChar(u8 loc,u8*pattern);

12- void LCD_PinsInit ();

2.4.9 Ultrasonic Module

1-void USCallBackFun(void);

2-u16 US_getdistance(void);

2.4.10 Motor Module

1- en_MotorError_t Car_Motors_init(void);

2- en_MotorError_t Car_Moving_FWD(void);

3- en_MotorError_t Car_Moving_BWD(void);

4- en_MotorError_t Car_Rotate_Right(void);

5- en_MotorError_t Car_Rotate_Left(void);

6- en_MotorError_t Car_Stop(void);

2.4.11 App Module

1- void app_init();

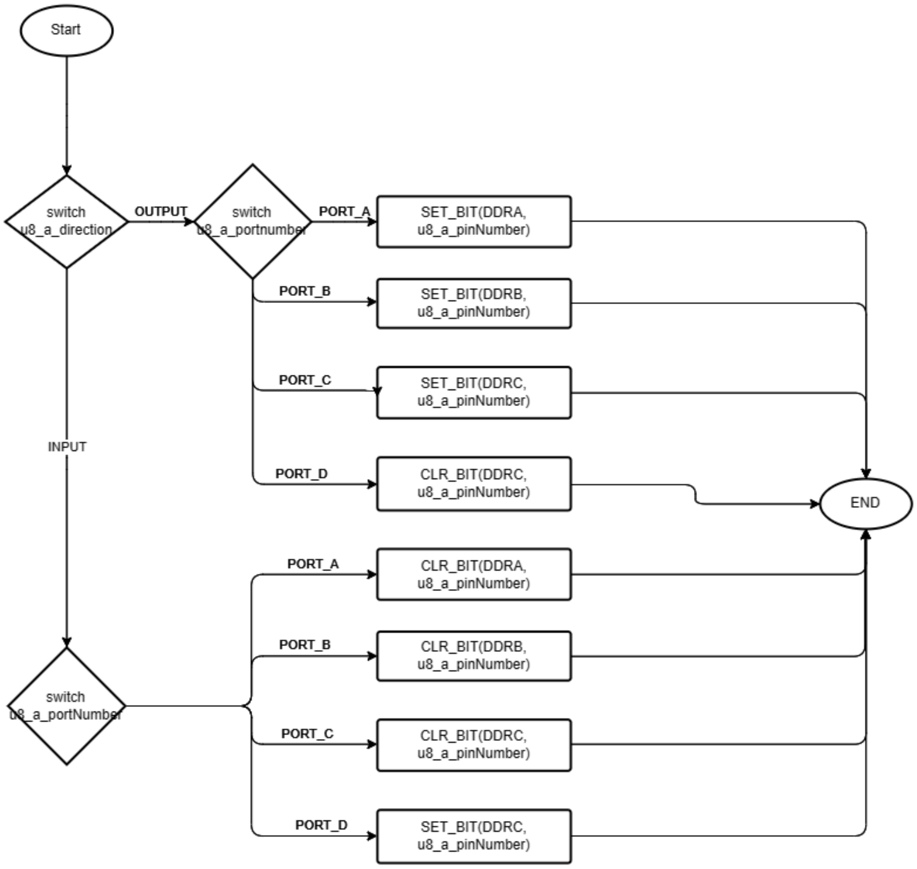
2- void app_start();

3-Low Level Design

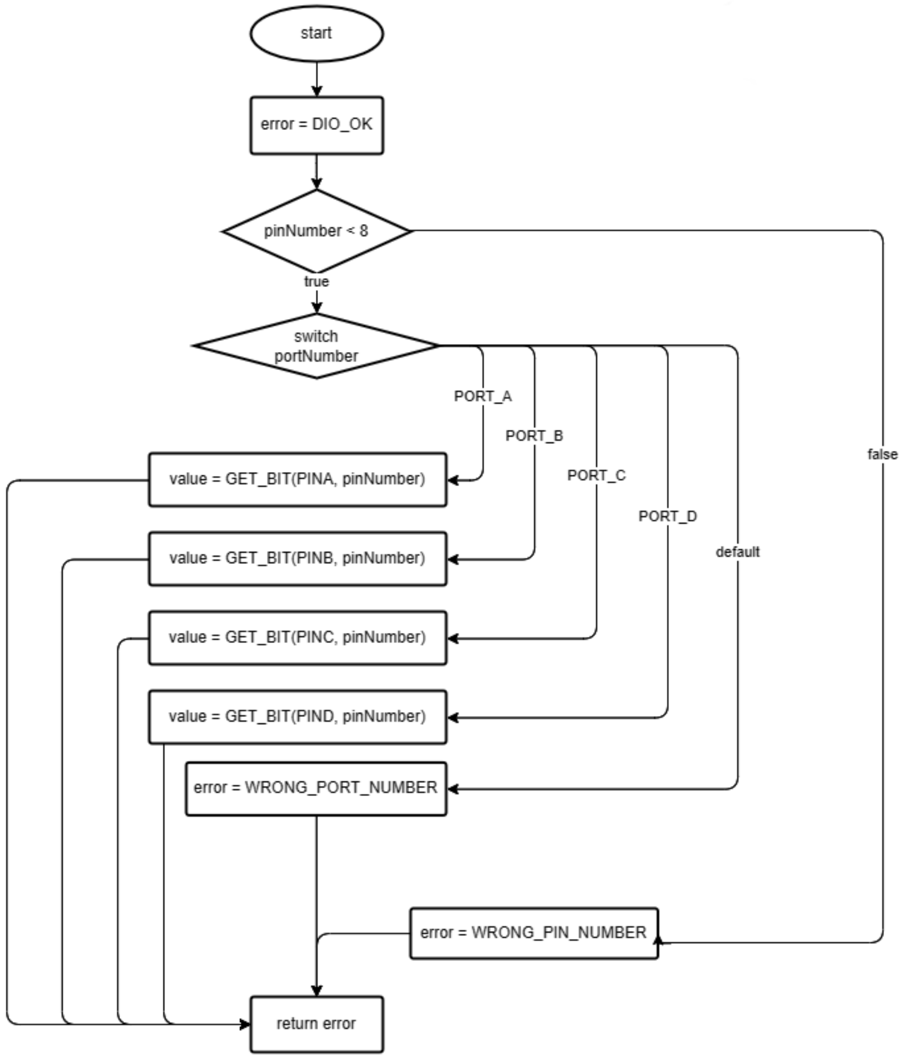
3.1 APIs Flow Chart

3.1.1 DIO API’s

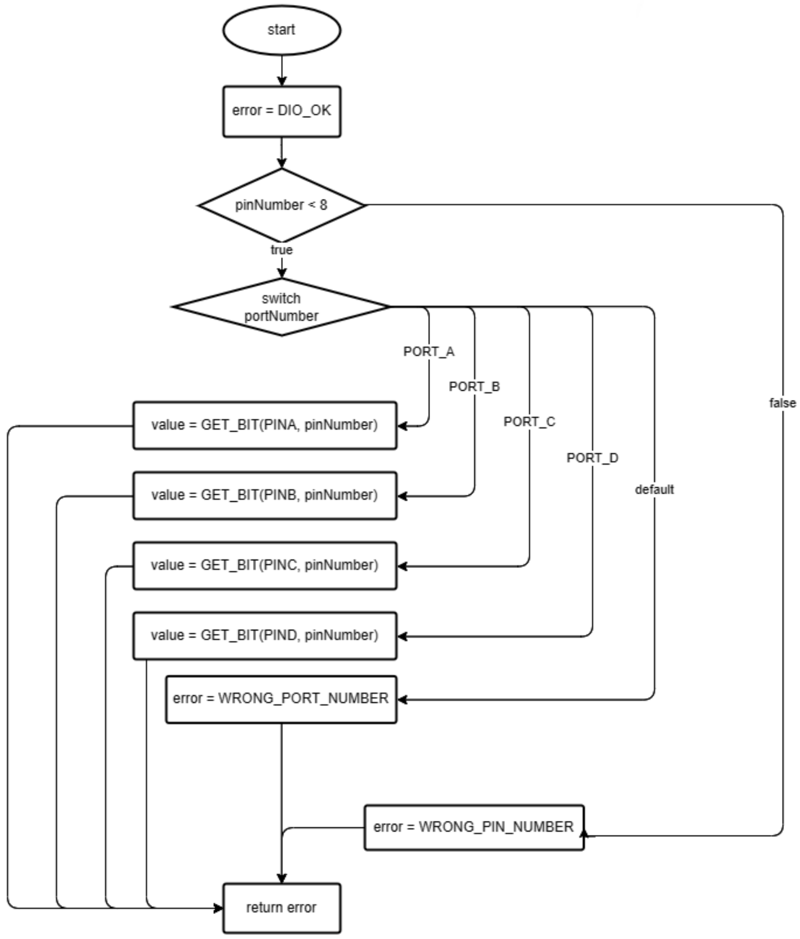
1- en_dioError_t DIO_initpin (DIO_Pin_type pin,DIO_PinStatus_type status)



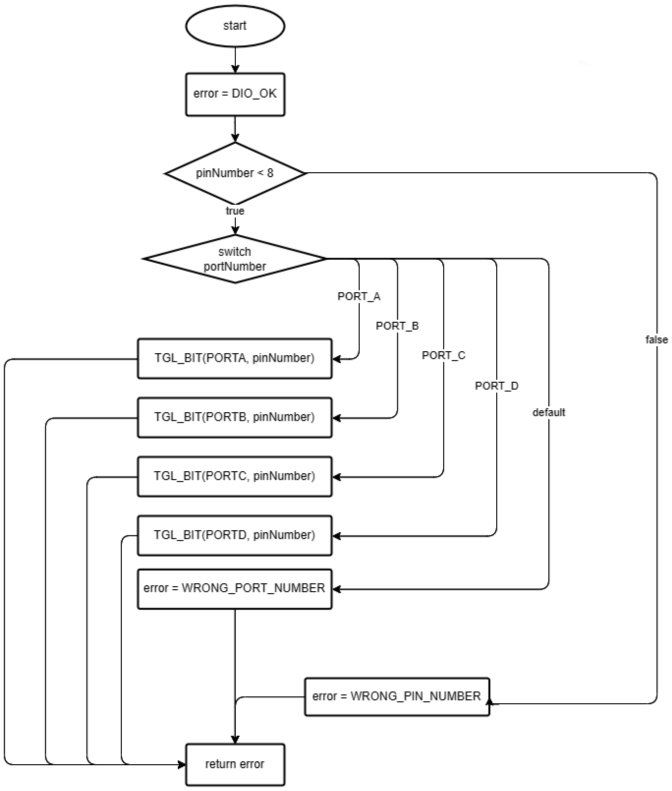
2- en_dioError_t DIO_writepin (DIO_Pin_type pin,DIO_PinVoltage_type volt)


start
error = DIO_OK
pinNumber < 8
true
false
switch portNumber
PORT_A
PORT_B
PORT_C
PORT_D
default
value = GET_BIT(PINA, pinNumber)
value = GET_BIT(PINB, pinNumber)
value = GET_BIT(PINC, pinNumber)
value = GET_BIT(PIND, pinNumber)
error = WRONG_PORT_NUMBER
error = WRONG_PIN_NUMBER
return error

3- en_dioError_t DIO_readpin (DIO_Pin_type pin,DIO_PinVoltage_type *volt)

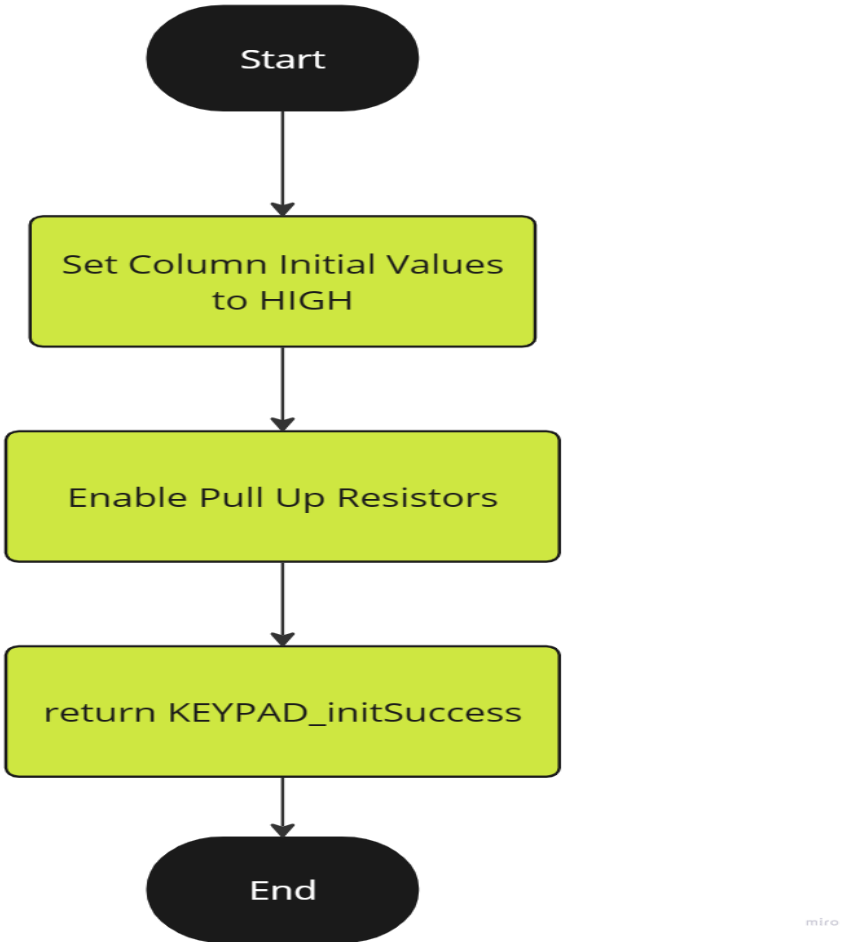

start
error = DIO_OK
pinNumber < 8
true
false
switch portNumber
PORT_A
PORT_B
PORT_C
PORT_D
default
value = GET_BIT(PINA, pinNumber)
value = GET_BIT(PINB, pinNumber)
value = GET_BIT(PINC, pinNumber)
value = GET_BIT(PIND, pinNumber)
error = WRONG_PORT_NUMBER
error = WRONG_PIN_NUMBER
return error

4- en_dioError_t DIO_togglepin(DIO_Pin_type pin)

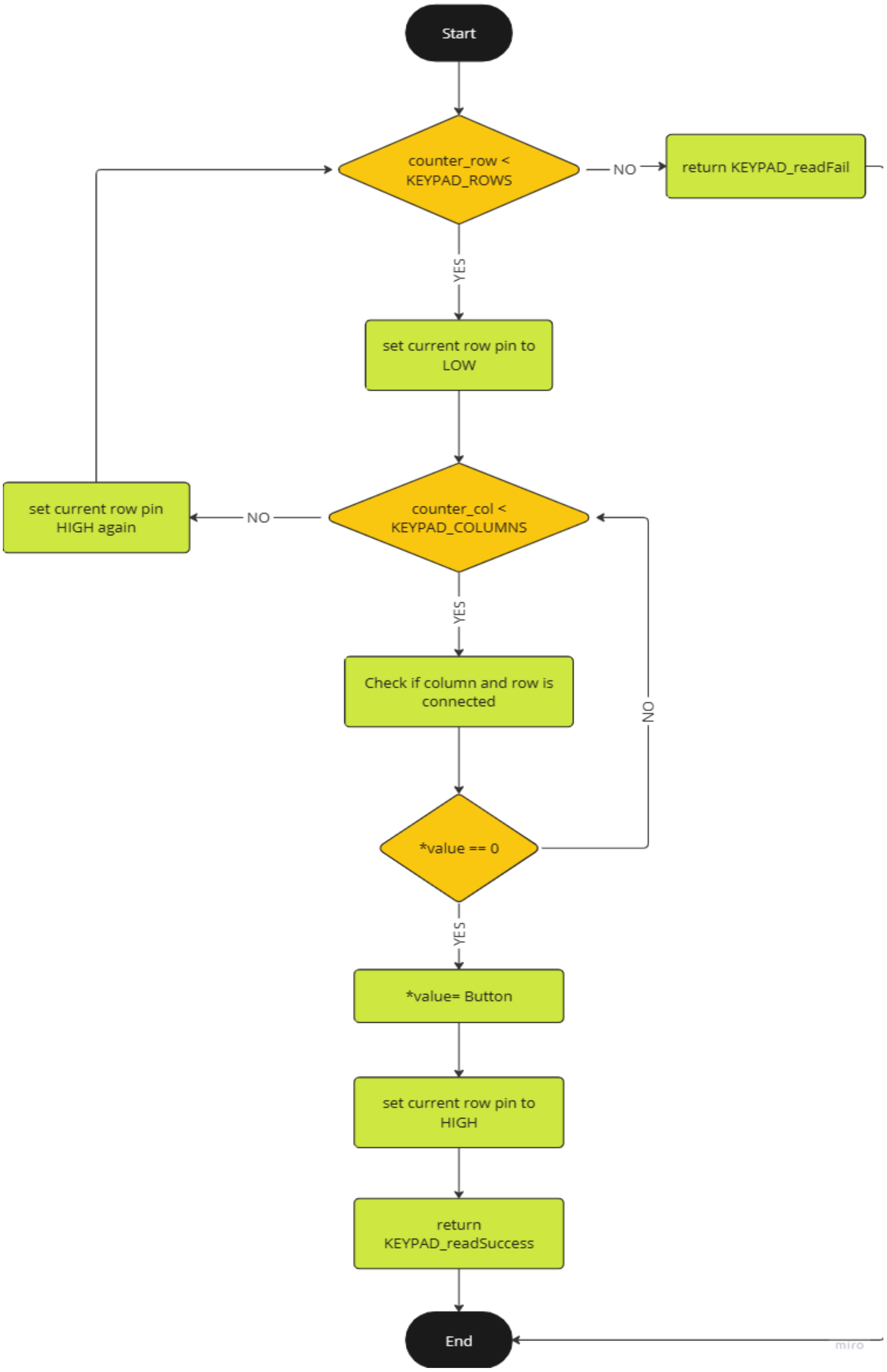

start
error = DIO_OK
pinNumber < 8
true
false
switch portNumber
PORT_A
PORT_B
PORT_C
PORT_D
default
TGL_BIT(PORTA, pinNumber)
TGL_BIT(PORTB, pinNumber)
TGL_BIT(PORTC, pinNumber)
TGL_BIT(PORTD, pinNumber)
error = WRONG_PORT_NUMBER
error = WRONG_PIN_NUMBER
return error

3.1. Keypad API's

KEYPAD_initError KEYPAD_init(void)

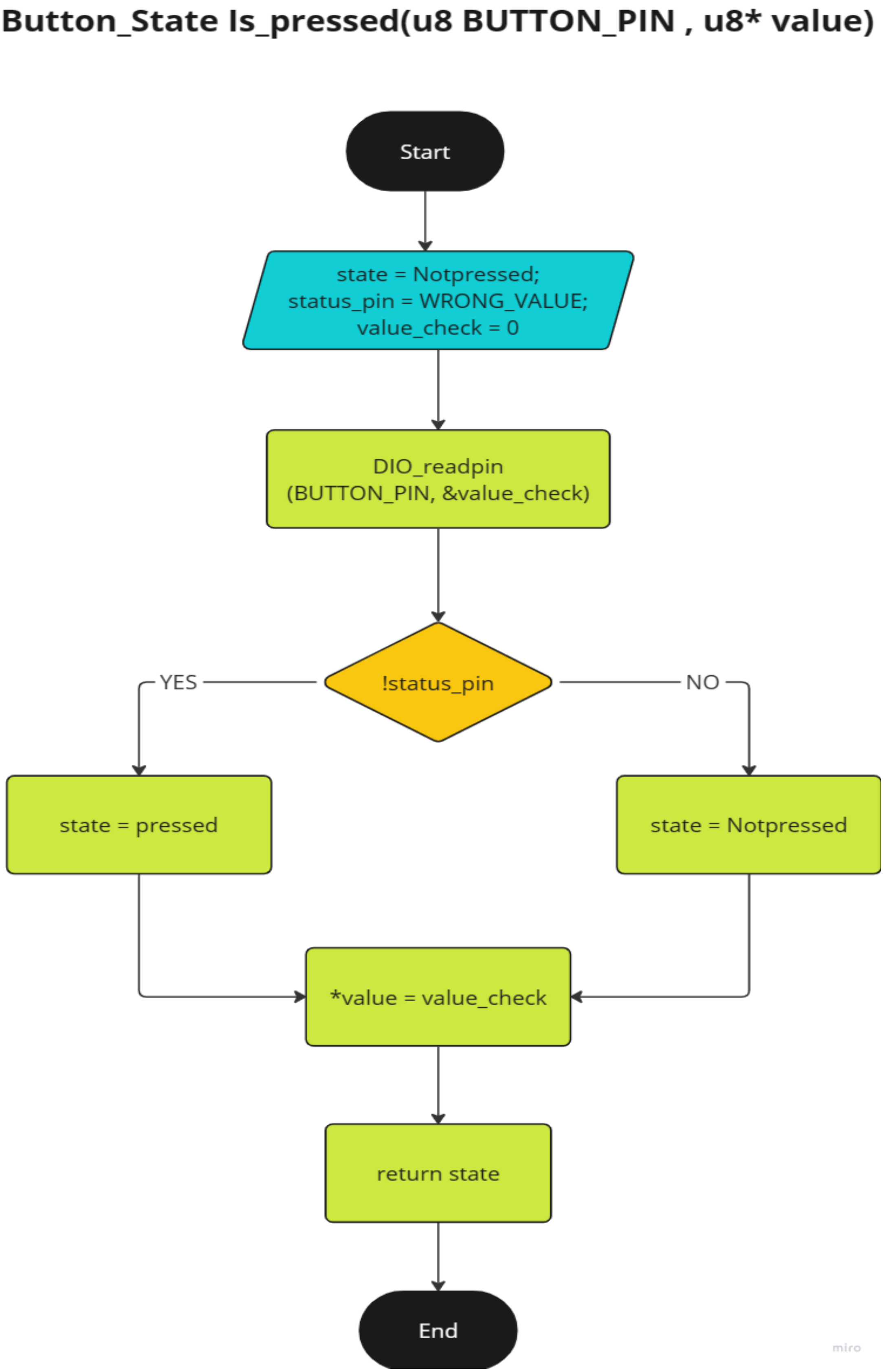


KEYPAD_readError KEYPAD_getpressedkey(u8 *value)



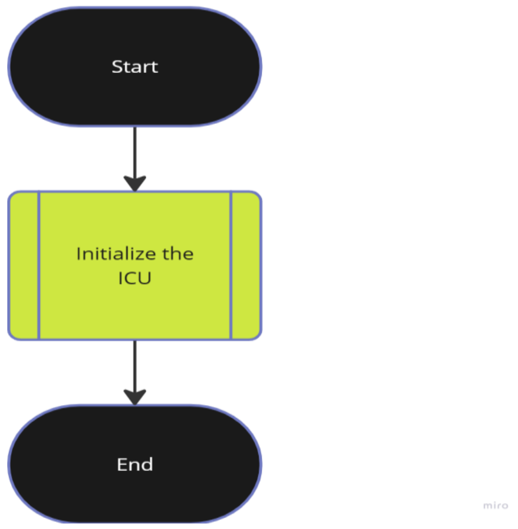
Start
counter_row < KEYPAD_ROWS
NO
return KEYPAD_readFail
YES
set current row pin to LOW
counter_col < KEYPAD_COLUMNS
NO
set current row pin HIGH again
YES
Check if column and row is connected
NO
*value == 0
YES
*value= Button
set current row pin to HIGH
return KEYPAD_readSuccess
End
miro

3.1. Button

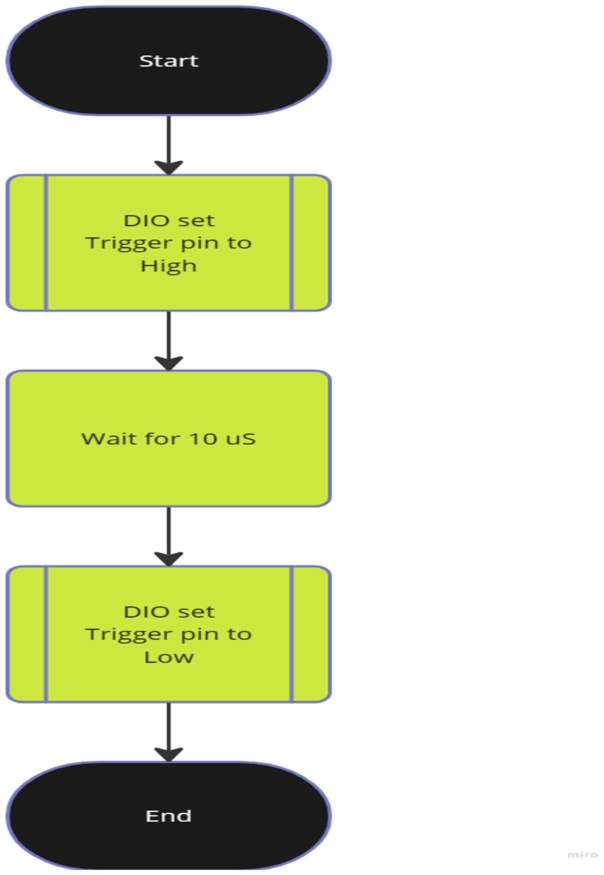

Button_State Is_pressed(u8 BUTTON_PIN , u8* value)
Start
state = Notpressed;
status_pin = WRONG_VALUE;
value_check = 0
DIO_readpin
(BUTTON_PIN, &value_check)
!status_pin
YES
NO
state = pressed
state = Notpressed
*value = value_check
return state
End
miro

3.1. Ultrasonic

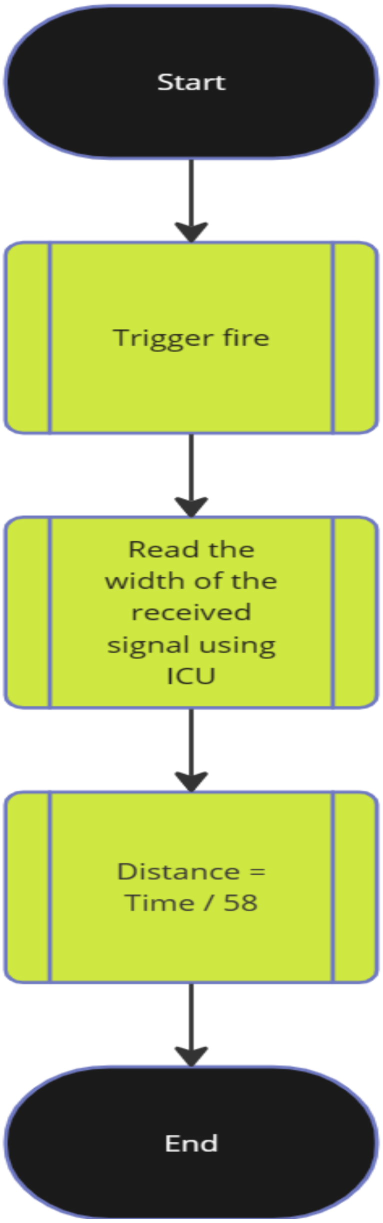
Ultrasonic module initialization



Ultrasonic module trigger fire

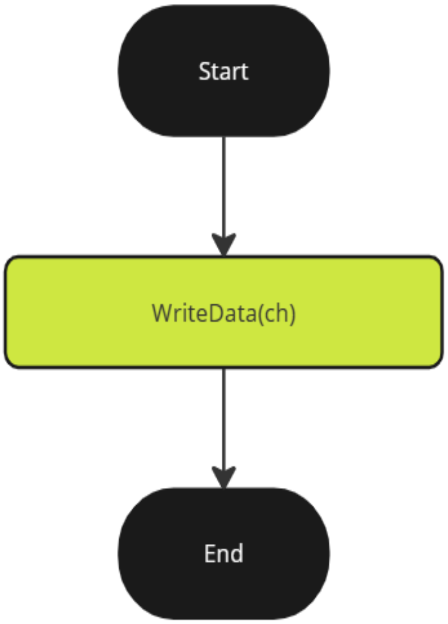


Ultrasonic module distance calculate

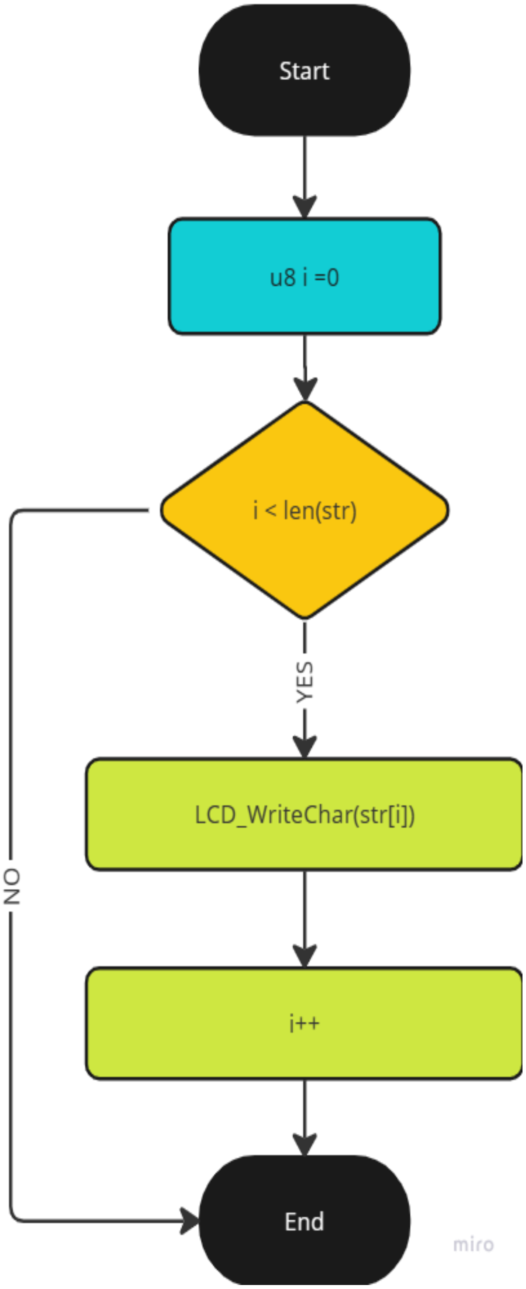


3.1. LCD

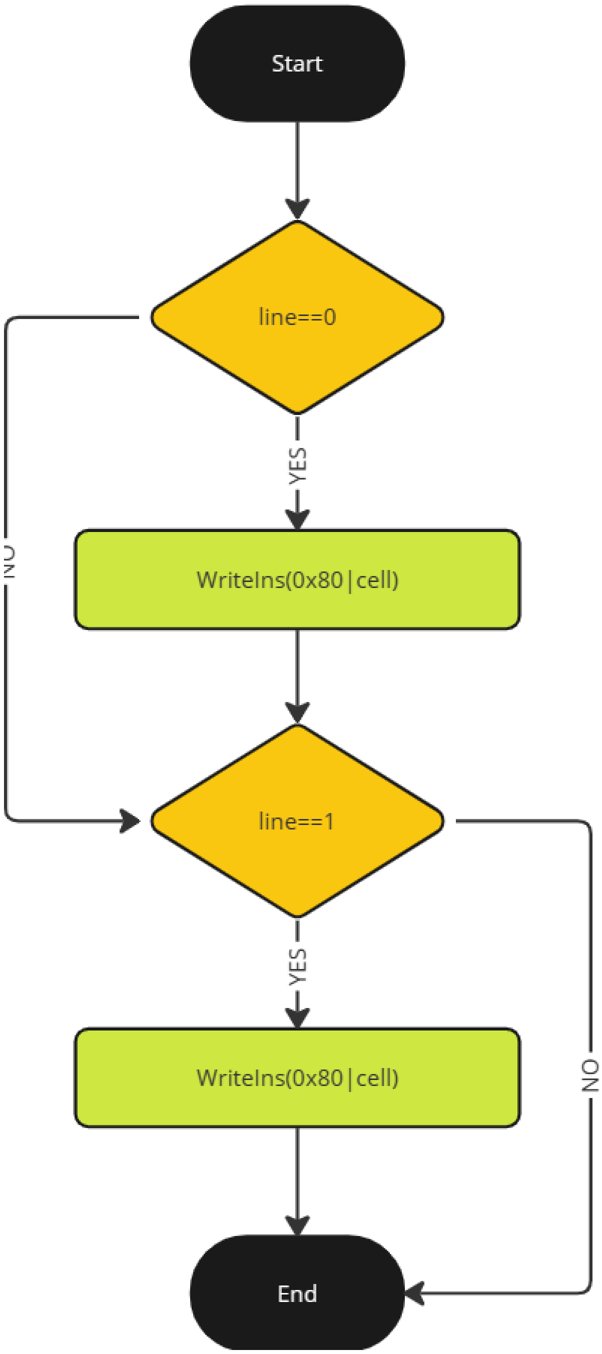
void LCD_WriteChar(u8 ch)



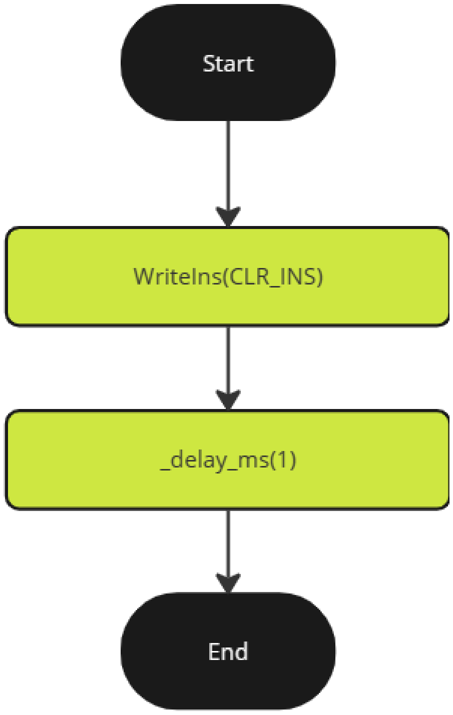
void LCD_WriteString(u8*str)



void LCD_SetCursor(u8 line,u8 cell)

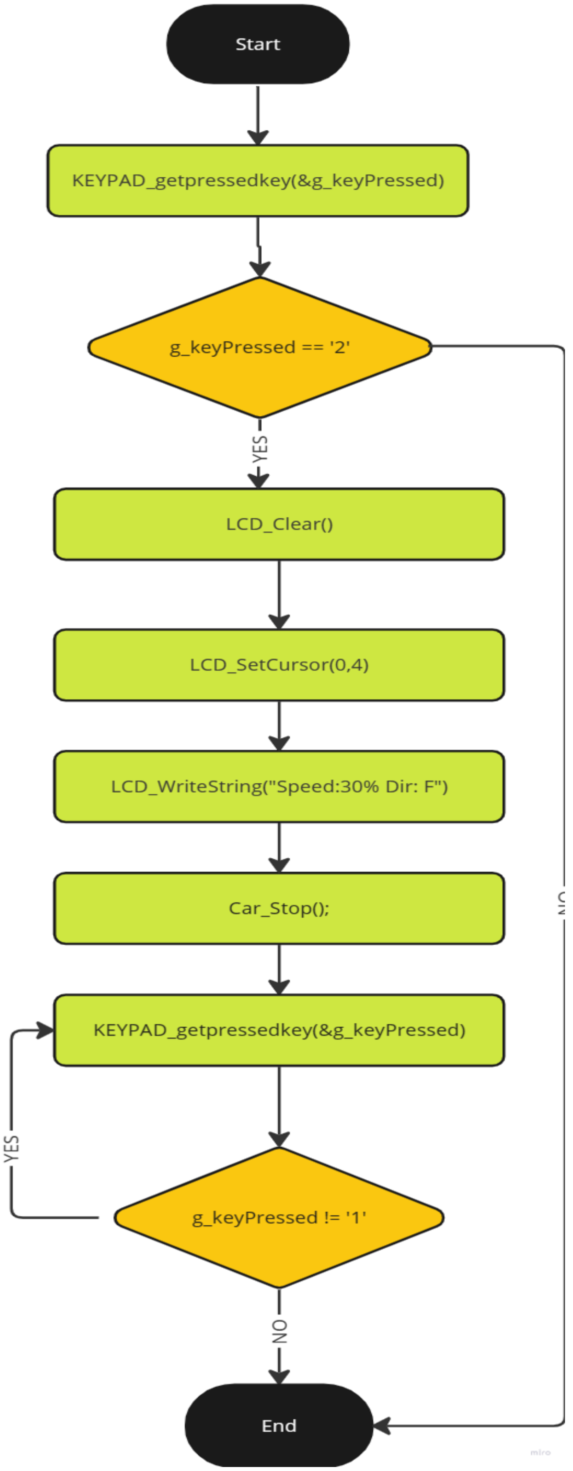


void LCD_Clear(void)

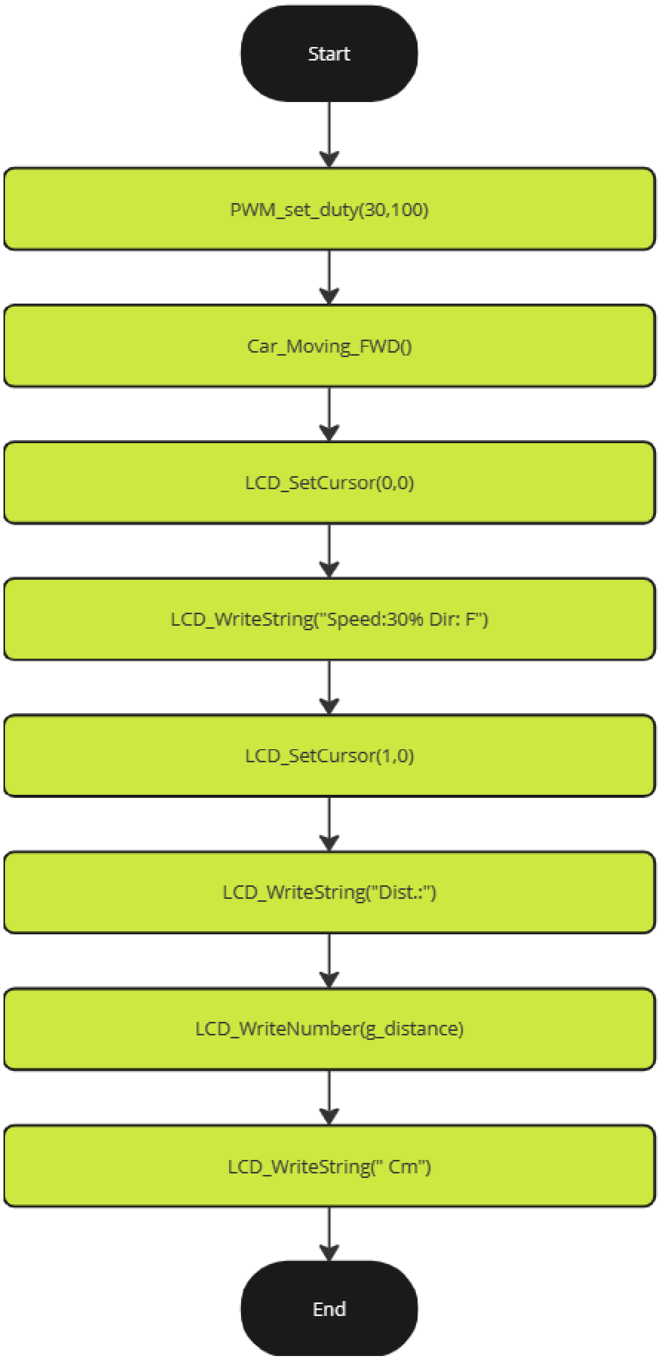


3.1. APP

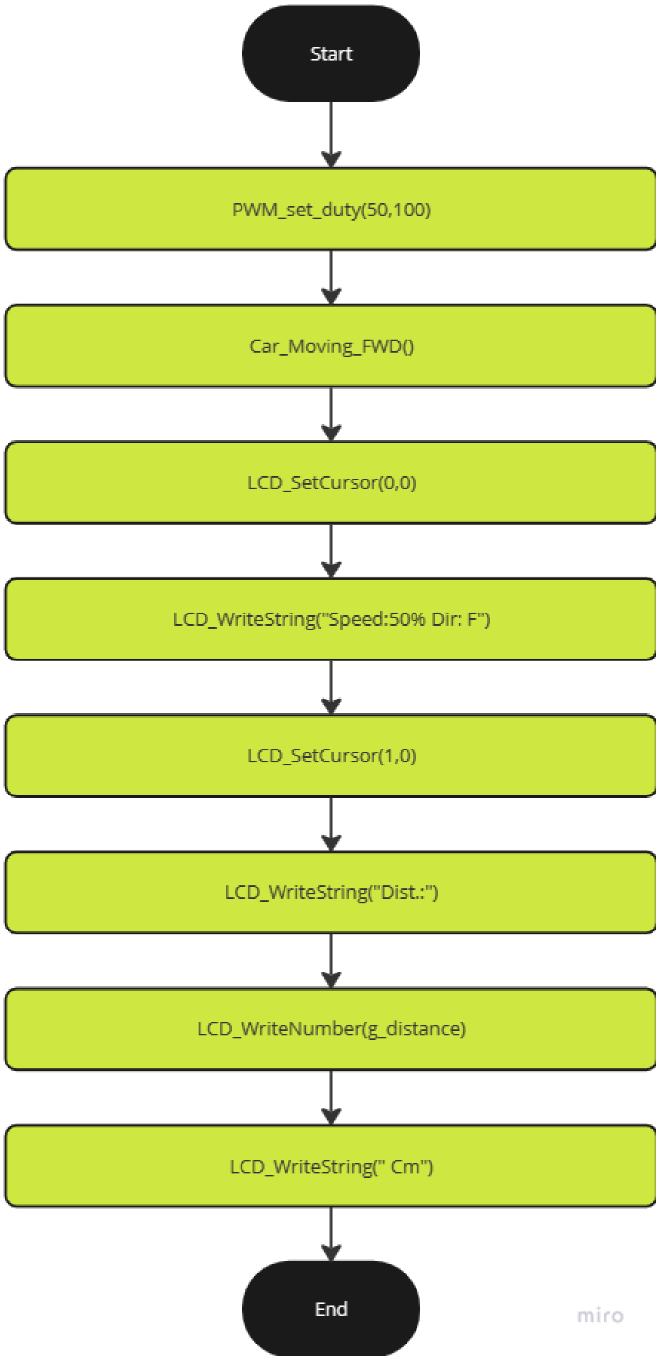
void STOP_check (void)


Start
KEYPAD_getpressedkey(&g_keyPressed)
g_keyPressed == '2'
YES
NO
LCD_Clear()
LCD_SetCursor(0,4)
LCD_WriteString("Speed:30% Dir: F")
Car_Stop();
KEYPAD_getpressedkey(&g_keyPressed)
YES
g_keyPressed != '1'
NO
End

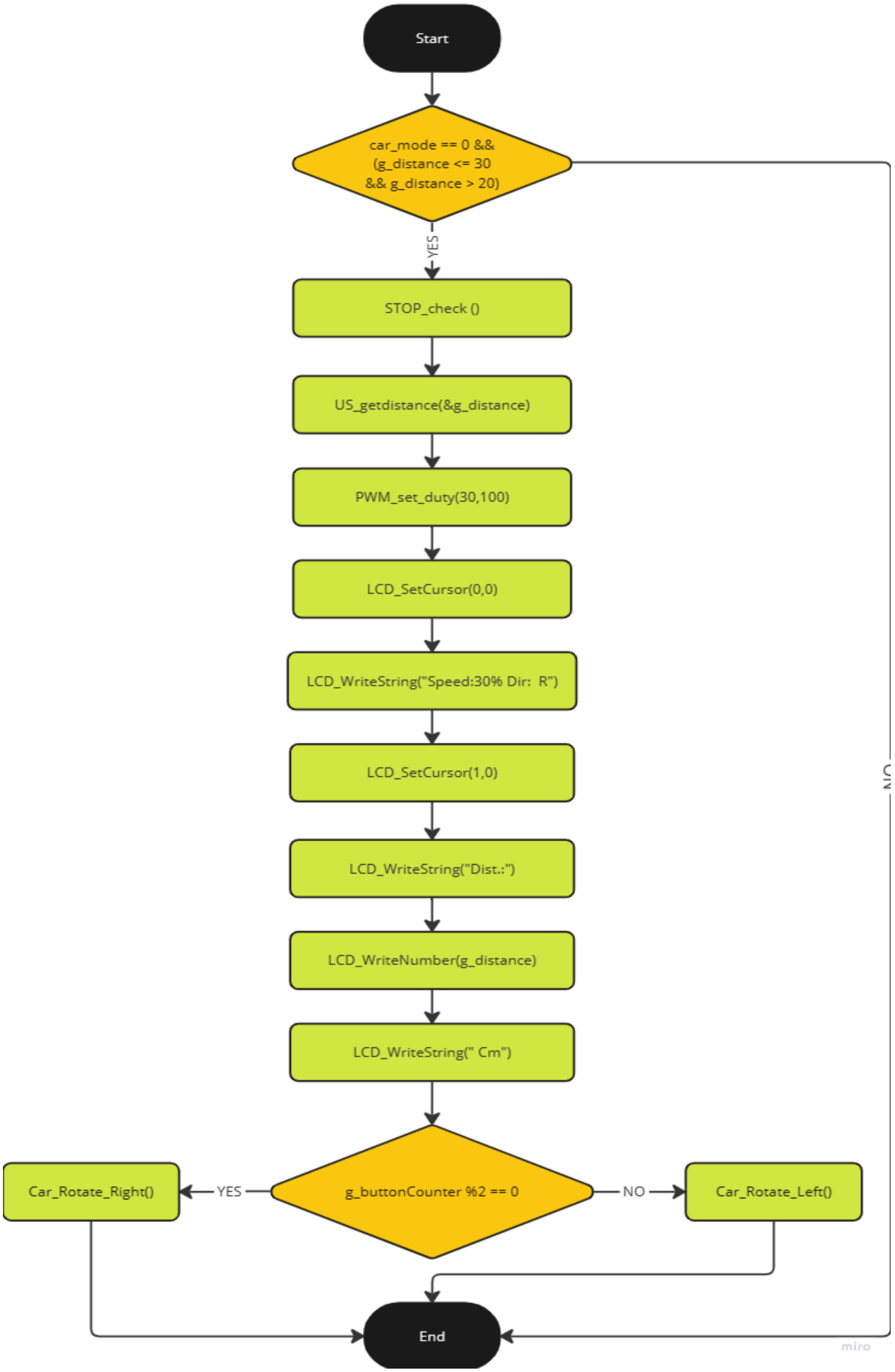
void car_Forward_30()



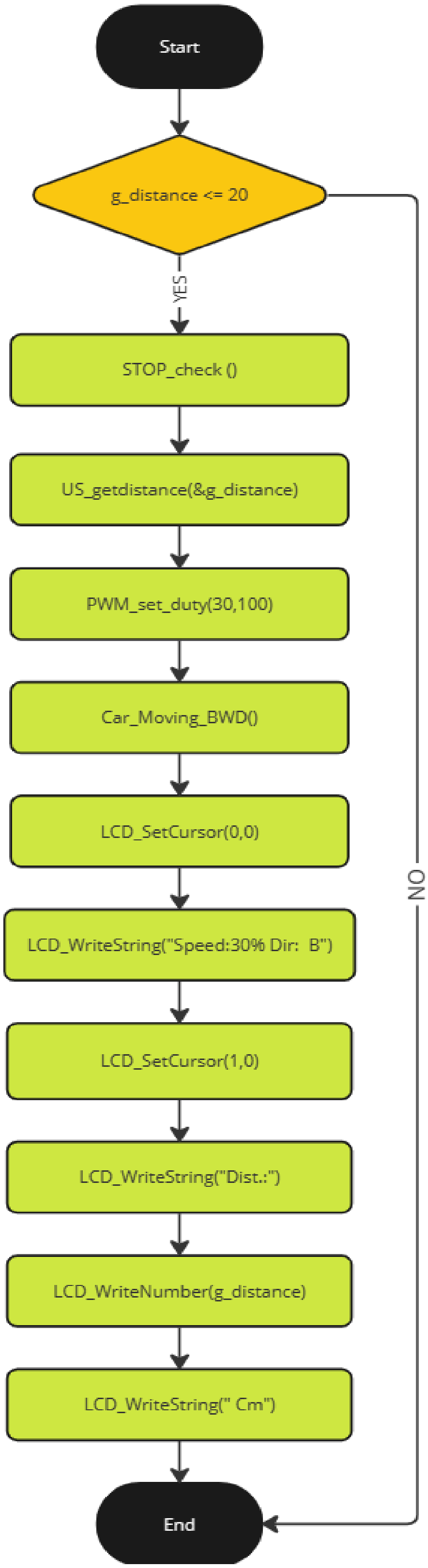
void car_Forward_50()



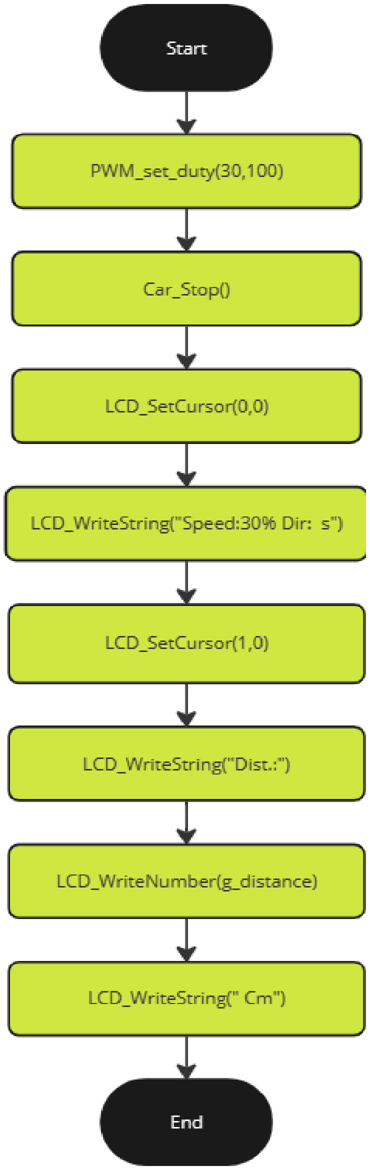
void car_Rotating()



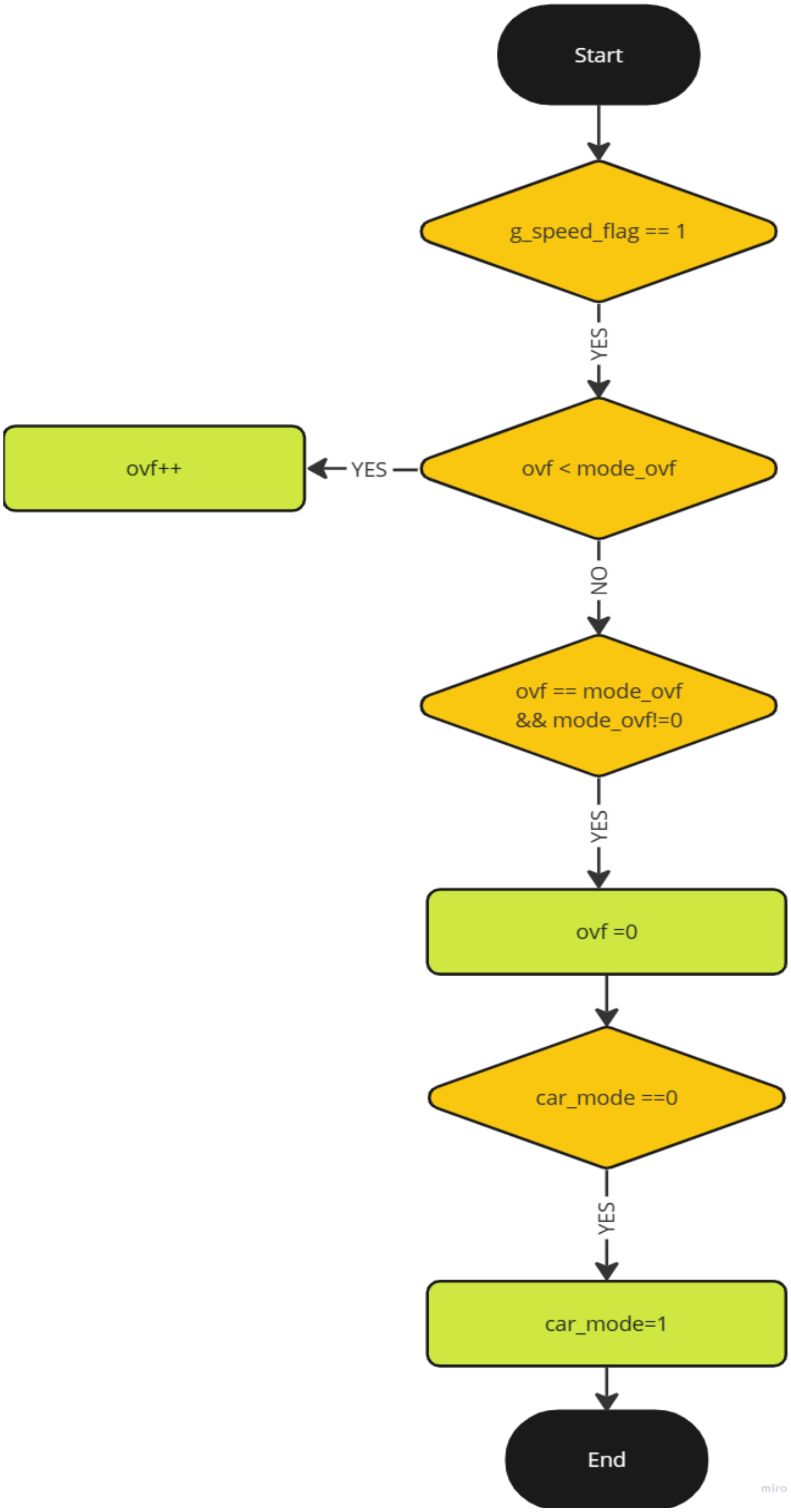
void car_Backword_30()



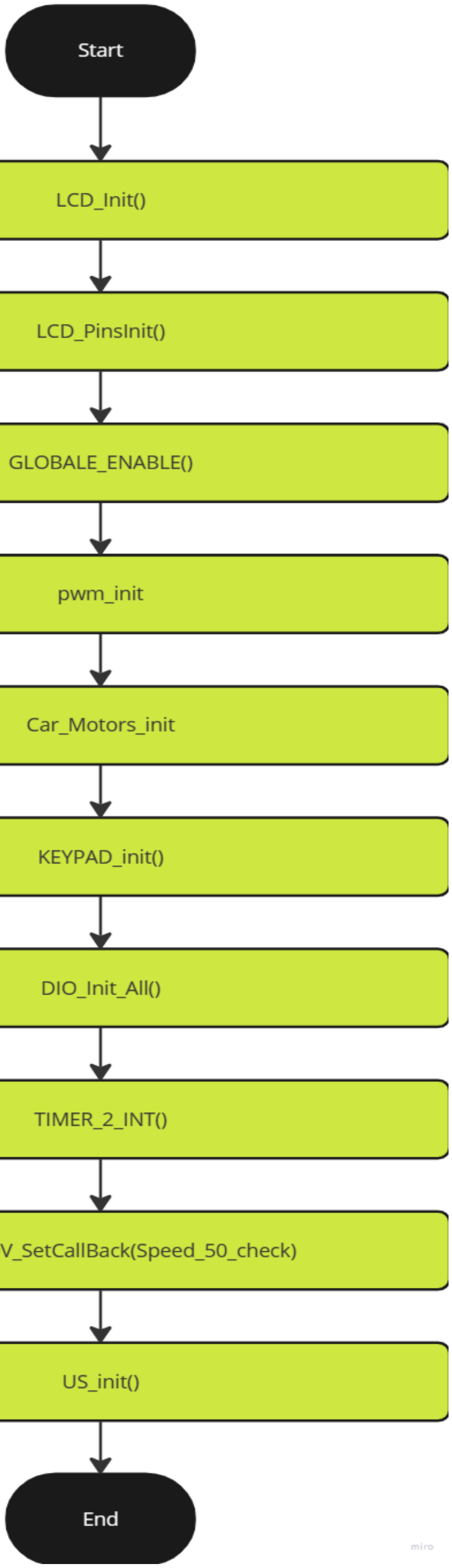
void Car_Stopping()



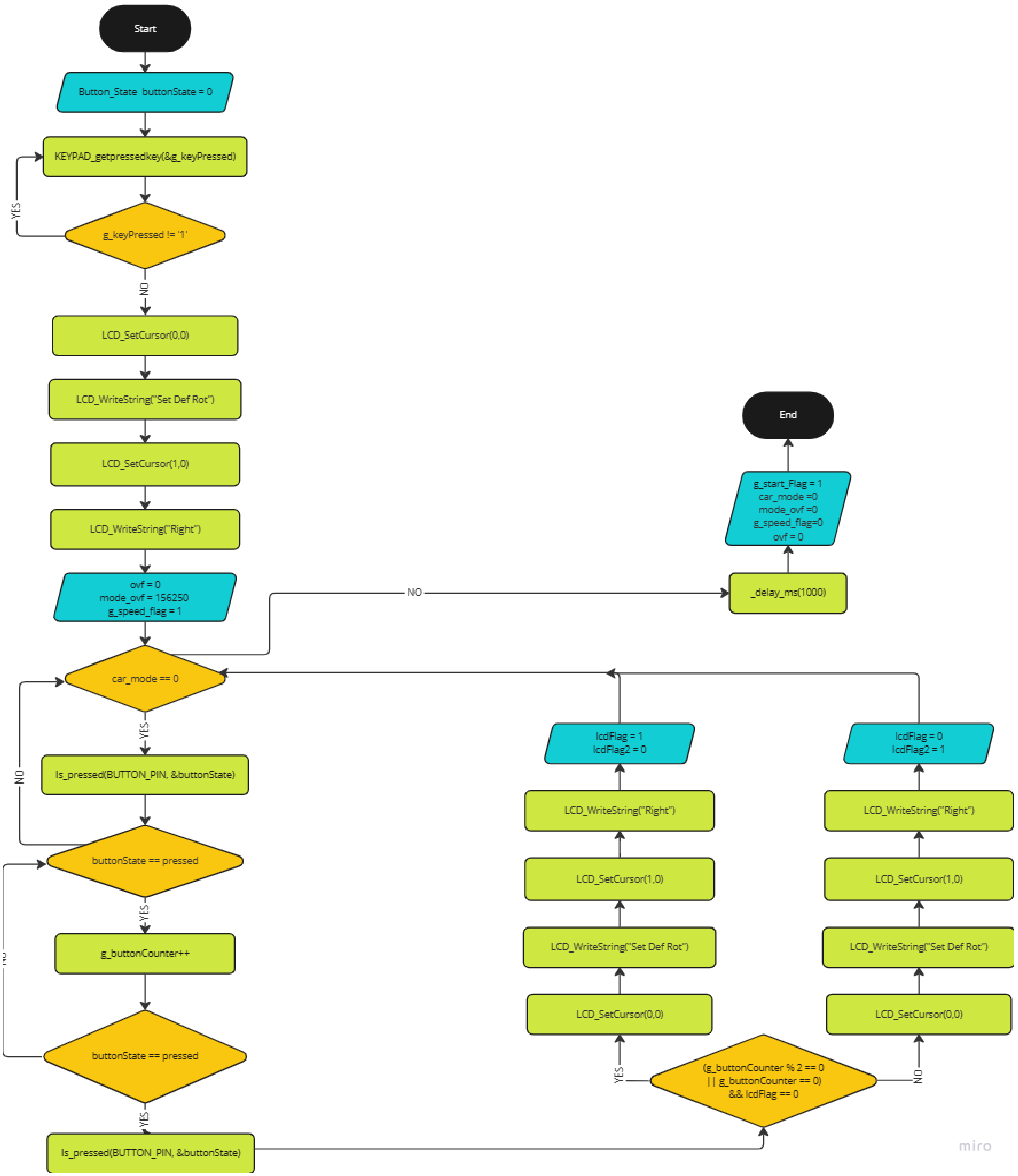
void Speed_50_check()



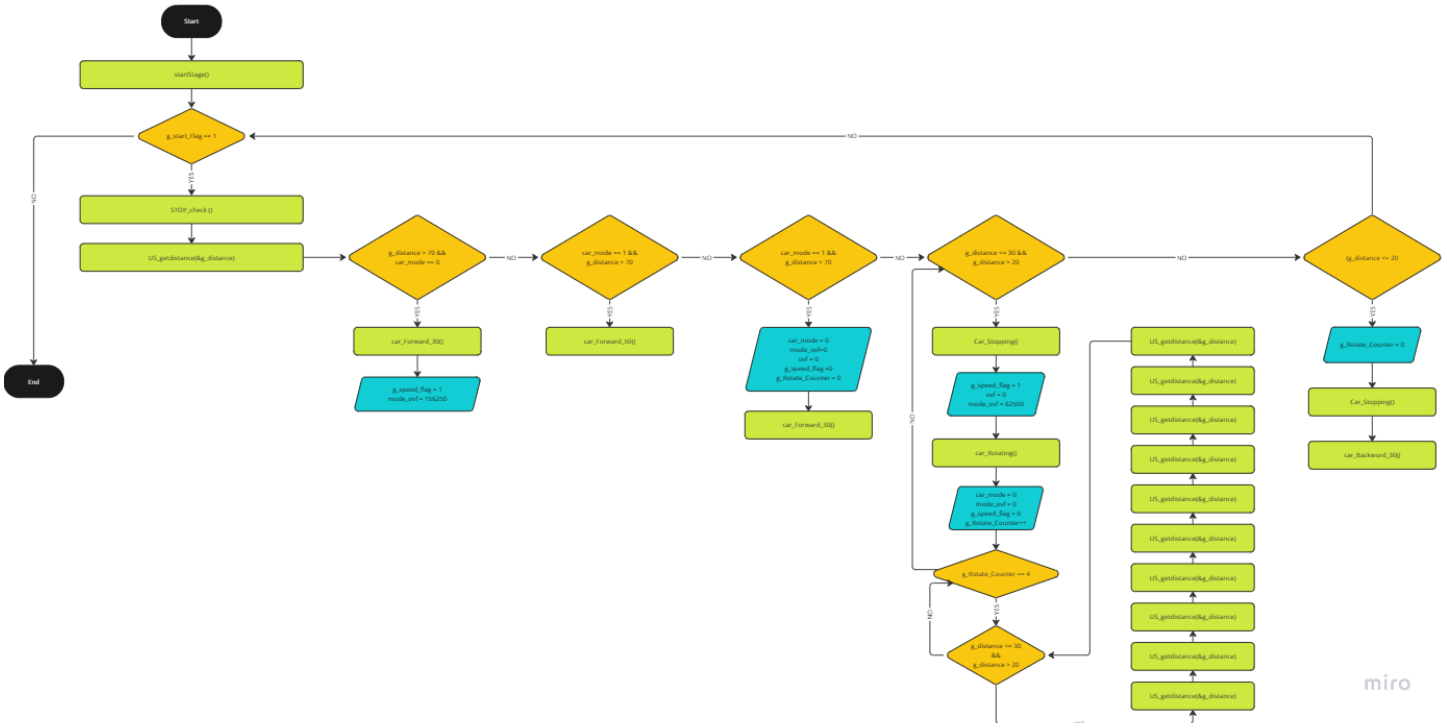
void app_init()



void startStage(void)

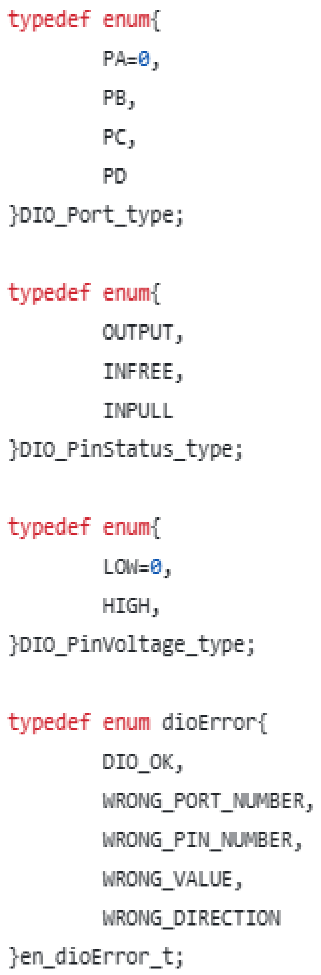


void app_start()

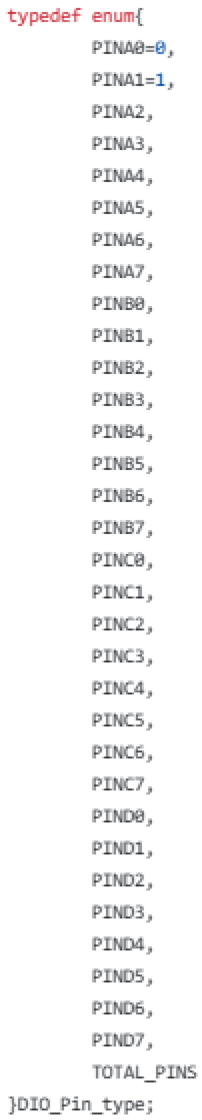


3.2 Precompiling & Linking Configurations

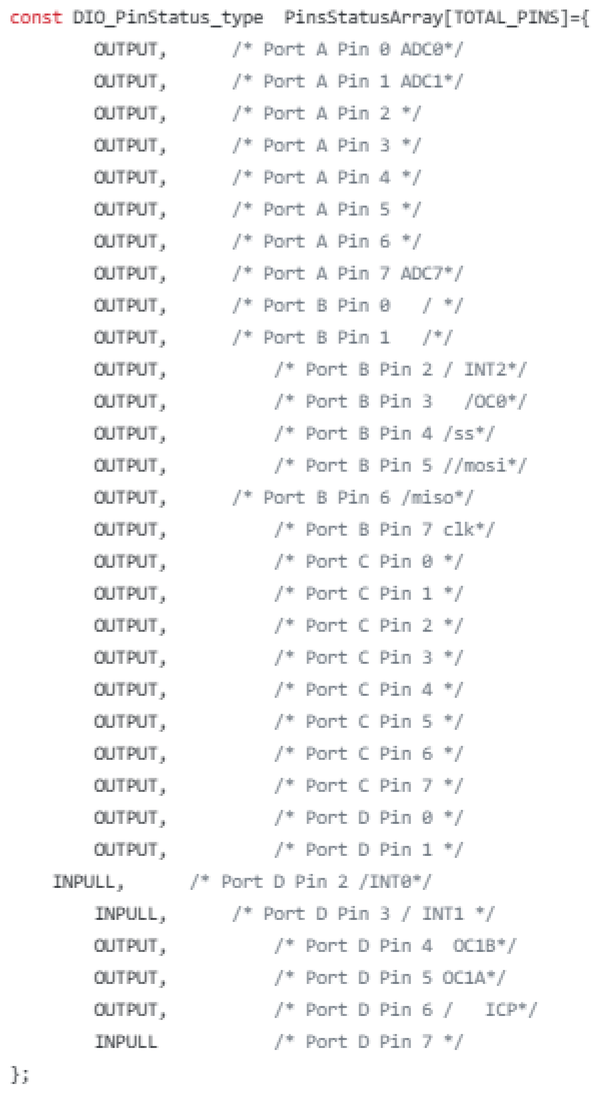
-DIO



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

typedef enum{
        OUTPUT,
        INFREE,
        INPULL
}DIO_PinStatus_type;

typedef enum{
        LOW=0,
        HIGH,
}DIO_PinVoltage_type;

typedef enum dioError{
        DIO_OK,
        WRONG_PORT_NUMBER,
        WRONG_PIN_NUMBER,
        WRONG_VALUE,
        WRONG_DIRECTION
}en_dioError_t;
```

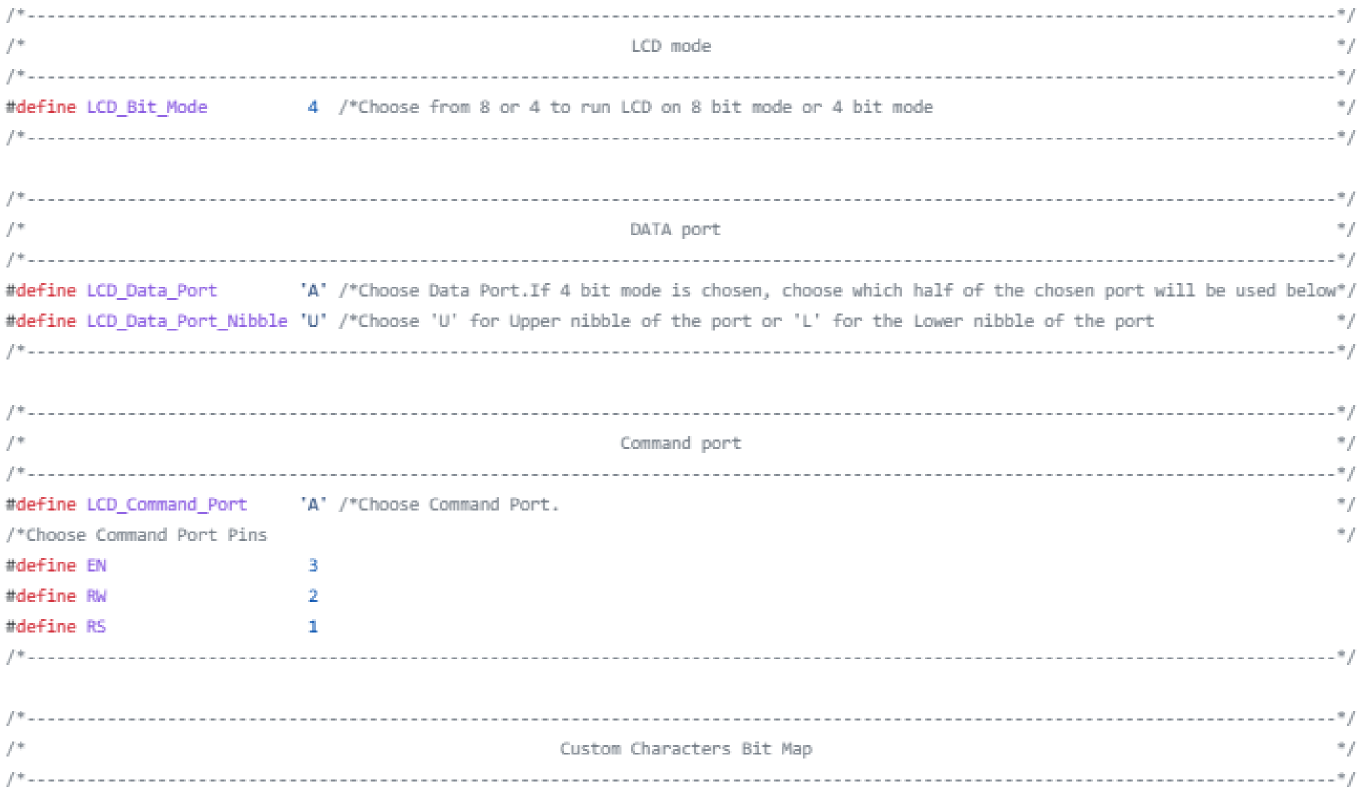


```
typedef enum{
        PINA0=0,
        PINA1=1,
        PINA2,
        PINA3,
        PINA4,
        PINA5,
        PINA6,
        PINA7,
        PINB0,
        PINB1,
        PINB2,
        PINB3,
        PINB4,
        PINB5,
        PINB6,
        PINB7,
        PINC0,
        PINC1,
        PINC2,
        PINC3,
        PINC4,
        PINC5,
        PINC6,
        PINC7,
        PIND0,
        PIND1,
        PIND2,
        PIND3,
        PIND4,
        PIND5,
        PIND6,
        PIND7,
        TOTAL_PINS
}DIO_Pin_type;
```



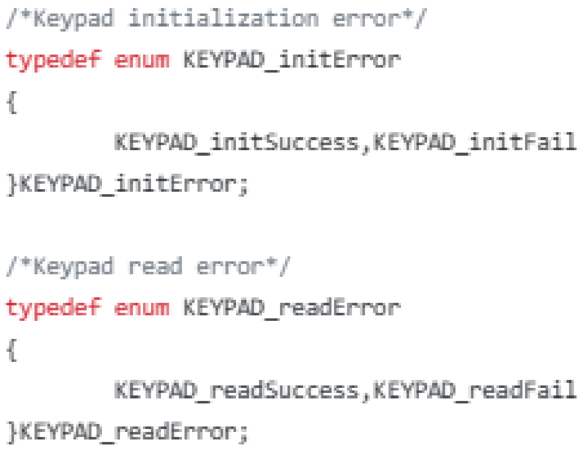
```
const DIO_PinStatus_type  PinsStatusArray[TOTAL_PINS]={
        OUTPUT,      /* Port A Pin 0 ADC0*/
        OUTPUT,      /* Port A Pin 1 ADC1*/
        OUTPUT,      /* Port A Pin 2 */
        OUTPUT,      /* Port A Pin 3 */
        OUTPUT,      /* Port A Pin 4 */
        OUTPUT,      /* Port A Pin 5 */
        OUTPUT,      /* Port A Pin 6 */
        OUTPUT,      /* Port A Pin 7 ADC7*/
        OUTPUT,      /* Port B Pin 0   / */
        OUTPUT,      /* Port B Pin 1   /*/
        OUTPUT,          /* Port B Pin 2 / INT2*/
        OUTPUT,          /* Port B Pin 3   /OC0*/
        OUTPUT,          /* Port B Pin 4 /ss*/
        OUTPUT,          /* Port B Pin 5 //mosi*/
        OUTPUT,      /* Port B Pin 6 /miso*/
        OUTPUT,          /* Port B Pin 7 clk*/
        OUTPUT,          /* Port C Pin 0 */
        OUTPUT,          /* Port C Pin 1 */
        OUTPUT,          /* Port C Pin 2 */
        OUTPUT,          /* Port C Pin 3 */
        OUTPUT,          /* Port C Pin 4 */
        OUTPUT,          /* Port C Pin 5 */
        OUTPUT,          /* Port C Pin 6 */
        OUTPUT,          /* Port C Pin 7 */
        OUTPUT,          /* Port D Pin 0 */
        OUTPUT,          /* Port D Pin 1 */
    INPULL,      /* Port D Pin 2 /INT0*/
        INPULL,      /* Port D Pin 3 / INT1 */
        OUTPUT,          /* Port D Pin 4  OC1B*/
        OUTPUT,          /* Port D Pin 5 OC1A*/
        OUTPUT,          /* Port D Pin 6 /   ICP*/
        INPULL           /* Port D Pin 7 */
};
```

-LCD

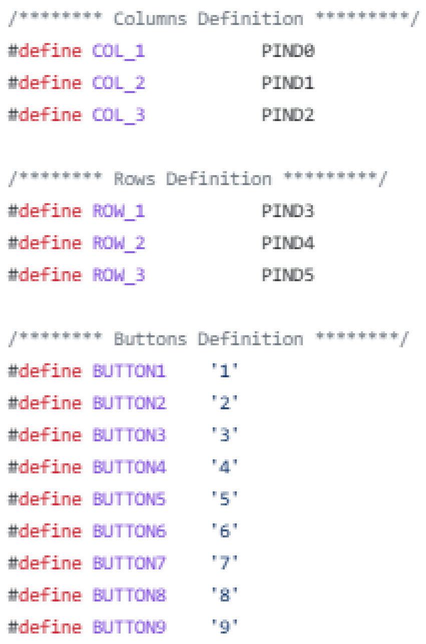


```
/*----------------------------------------------------------------------------------------------------------------------------------*/
/*                                                            LCD mode                                                              */
/*----------------------------------------------------------------------------------------------------------------------------------*/
#define LCD_Bit_Mode          4  /*Choose from 8 or 4 to run LCD on 8 bit mode or 4 bit mode                                        */
/*----------------------------------------------------------------------------------------------------------------------------------*/

/*----------------------------------------------------------------------------------------------------------------------------------*/
/*                                                            DATA port                                                             */
/*----------------------------------------------------------------------------------------------------------------------------------*/
#define LCD_Data_Port        'A' /*Choose Data Port.If 4 bit mode is chosen, choose which half of the chosen port will be used below*/
#define LCD_Data_Port_Nibble 'U' /*Choose 'U' for Upper nibble of the port or 'L' for the Lower nibble of the port                  */
/*----------------------------------------------------------------------------------------------------------------------------------*/

/*----------------------------------------------------------------------------------------------------------------------------------*/
/*                                                           Command port                                                           */
/*----------------------------------------------------------------------------------------------------------------------------------*/
#define LCD_Command_Port     'A' /*Choose Command Port.                                                                             */
/*Choose Command Port Pins                                                                                                          */
#define EN                    3
#define RW                    2
#define RS                    1
/*----------------------------------------------------------------------------------------------------------------------------------*/

/*----------------------------------------------------------------------------------------------------------------------------------*/
/*                                                     Custom Characters Bit Map                                                    */
/*----------------------------------------------------------------------------------------------------------------------------------*/
```

-Keypad



```
/*Keypad initialization error*/
typedef enum KEYPAD_initError
{
        KEYPAD_initSuccess,KEYPAD_initFail
}KEYPAD_initError;

/*Keypad read error*/
typedef enum KEYPAD_readError
{
        KEYPAD_readSuccess,KEYPAD_readFail
}KEYPAD_readError;
```



```
/******** Columns Definition *********/
#define COL_1           PIND0
#define COL_2           PIND1
#define COL_3           PIND2

/******** Rows Definition *********/
#define ROW_1           PIND3
#define ROW_2           PIND4
#define ROW_3           PIND5

/******** Buttons Definition ********/
#define BUTTON1    '1'
#define BUTTON2    '2'
#define BUTTON3    '3'
#define BUTTON4    '4'
#define BUTTON5    '5'
#define BUTTON6    '6'
#define BUTTON7    '7'
#define BUTTON8    '8'
#define BUTTON9    '9'
```