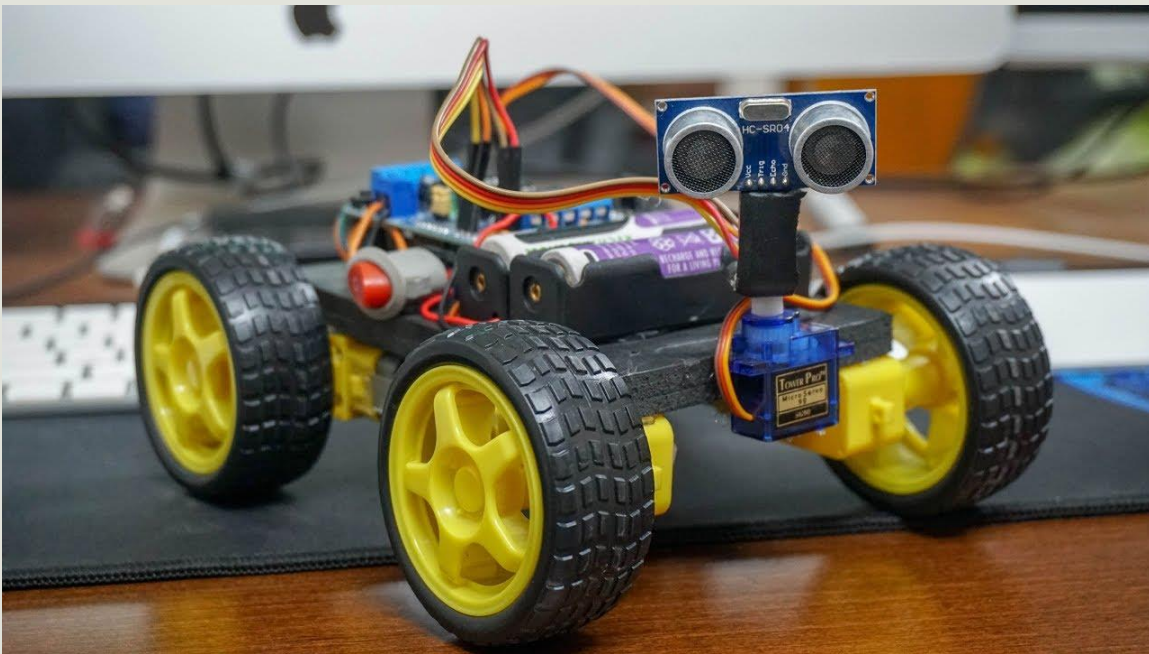


Obstacle Avoidance Robot V1.0



By: Nada Abdelazim Mohamed

Table of contents

1. Table of content
2. Project introduction
3. High Level Design
 1. Layered architecture
 2. Modules Descriptions
 3. Drivers' documentation
4. Low Level Design
 1. Provide the flowchart for each function in each module.
 2. Pre-compiling configurations for each module
 3. Linking configurations for each module

Project Introduction

This is a document to design a four-wheeled robot that avoids any obstacle in front of it.

After setting the default direction of rotation and waiting for five seconds, the robot moves with varying speeds based on the distance from any detected obstacle. The LCD displays information about the direction of movement, distance, and status of the robot.

Hardware Components:

The components are: ATmega32 microcontroller, four motors, one ultrasonic sensor, push-button, a keypad with two buttons and LCD.

Ultrasonic Sensor Connections:

The sensory component of the robot features Trig connected to Port B pin3 and Echo connected to Port B pin2.

System Requirements:

The robot starts at 0 speed.

The default rotation direction is to the right.

Pressing the first button on the keypad starts the robot, and the second button stops it.

After pressing start, the LCD displays a message for selecting the default rotation direction of the robot.

The robot waits for five seconds to choose between left and right.

The LCD displays speed and moving direction, as well as the object distance. Movement directions include forward, backwards, rotating, or stopped.

Default rotation direction changes based on pressing PBUTTON0 once or twice.

After five seconds, the robot starts moving at 30% speed.

Robot's movement:

Robot Start up movement:

The robot initiates movement two seconds after the default direction of rotation is set. The LCD displays a welcome message of “Set Def Rot” when the robot starts.

Robot Movement with no Obstacles or obstacles at distance more 70 cm:

The robot moves forward at 30% speed for the first five seconds. After that, it moves with 50% of the speed unless there is an obstacle at a distance less than 70 centimeters.

Robot Movement with Obstacles at 70 to 30 cm distance:

If there is an obstacle at a distance between 30 and 70 centimeters, the robot decreases its speed to 30%, and the LCD displays this change.

Robot Movement with Obstacles at 20 to 30 cm distance:

When the robot detects an obstacle between 20 and 30 centimeters, it stops and rotates 90 degrees either to the right or left, based on the chosen configuration. This change in direction is displayed on the LCD.

Robot Movement with Obstacles at less than 20 cm distance:

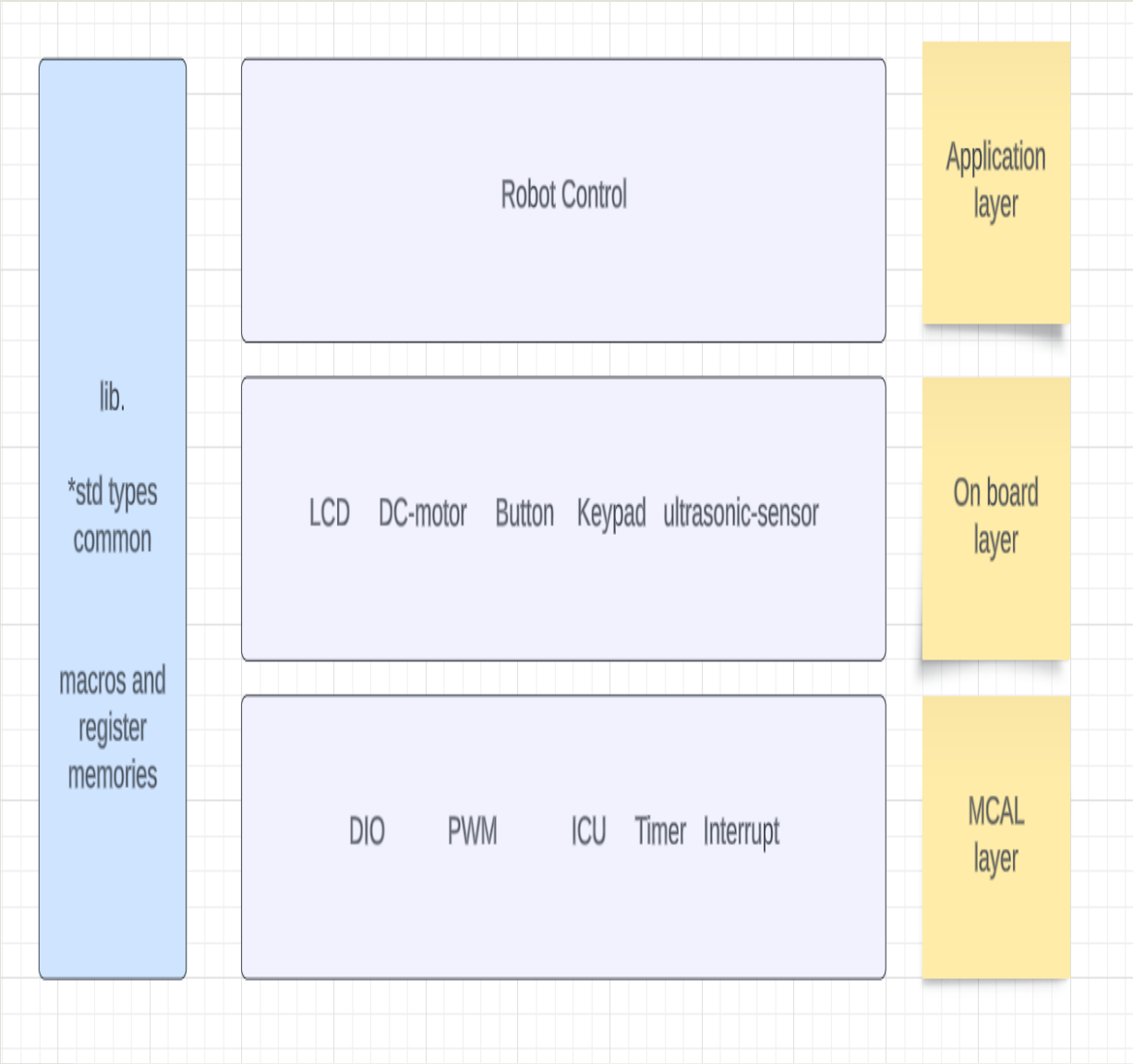
If there is an obstacle at a distance between 0 and 20 centimeters, the robot stops and moves backwards with 30% speed until it is beyond 20 cm to 30 cm distance.

Obstacle detection:

If the robot rotates 360 degrees without detecting distances greater than 20 centimeters, it stops. The LCD displays information that the robot has stopped. The robot checks if an obstacle was removed every three seconds and moves in the direction of the furthest object.

HIGH LEVEL DESIGN

Layered architecture



Layers and Modules descriptions

Layers description:

(1)- Application layer:

Contains functions calls to implement the main project.

(2)- On Board layer:

Contains Drivers of the external electronic devices which will be connected to the microcontroller and the system overall.

(3)-MCAL: "Microcontroller Abstraction Layer"

Contains interfaces of the microcontroller's peripherals.

Modules descriptions:

1. DIO module: Digital Input/Output module is used to control and read digital signals from external devices.
2. PWM: Pulse Width Modulation module is used to generate analog signals by varying the duty cycle of a digital signal. In this project the PWM module is used to control the speed of the motors.
3. ICU module: Input Capture Unit module is used to measure the time between two events or pulses.
4. Timer module: Timer module is used to generate precise time delays or periodic interrupts.
5. Interrupt module: Interrupt module is used to handle external events or interrupts that occur during program execution.
6. LCD module: Liquid Crystal Display module is used to display text and graphics on a screen.
7. Dc motor module: DC Motor module is used to control the speed and direction of a DC motor.
8. Keypad module: Keypad is used to read input from a matrix of buttons arranged in rows and columns.
9. Button: Button module is used to detect when a button has been pressed or released.

10. Ultrasonic sensor module: Uses an ultrasonic sensor to detect the location and distance of objects within the robot's surroundings.

Driver's documentation

1. DIO module APIs:

```
void DIO_init(uint8_t pinNumber, uint8_t portNumber, uint8_t direction);
```

```
void DIO_write(uint8_t pinNumber, uint8_t portNumber, uint8_t value);
```

```
void DIO_toggle(uint8_t pinNumber, uint8_t portNumber);
```

```
void DIO_read(uint8_t pinNumber, uint8_t portNumber, uint8_t *value);
```

2. PWM module APIs:

```
void TIMER_0_pwm(float a_intial);
```

3. ICU module APIs:

```
void ICU_interrupt_trigger(void);
```

4. Timer module APIs:

```
Timer_ErrorStatus TIMER_0_init(Timer_Mode a_mode);
```

```
Timer_ErrorStatus TIMER_0_start(Timer_Prescaler a_prescaler);
```

```
void TIMER_0_stop(void);
```

```
Timer_ErrorStatus TIMER_0_setIntialValue(double a_value);
```

```
Timer_ErrorStatus TIMER_0_OvfNum(double a_overflow);
```

```
void TIMER_0_DELAY_MS(double a_time_ms);
```

```
Timer_ErrorStatus TIMER_2_init(Timer_Mode a_mode);
```

```
Timer_ErrorStatus TIMER_2_start(Timer_Prescaler a_prescaler);
```

```
void TIMER_2_stop(void);
```

```
Timer_ErrorStatus TIMER_2_setIntialValue(uint8_t a_value);
```

```
Timer_ErrorStatus TIMER_2_OvfNum(double overflow);  
  
void TIMER_2_DELAY_MS(double _delay);
```

5. Interrupt module APIs:

```
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));
```

6. LCD module APIs:

```
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_Character(uint8_t a_char);  
  
void LCD_Create_Character(uint8_t*a_Pattern, uint8_t a_Adress);
```

7. Dc motor module APIs:

```
Std_ReturnType dc_motor_init(const dc_motor_t *_dc_motor);  
  
std_ReturnType dc_motor_right(const dc_motor_t *_dc_motor);  
  
std_ReturnType dc_motor_stop(const dc_motor_t *_dc_motor);
```

8. Keypad module APIs:

```
void keypad_init (void);
```

```
void keypad_get_value (uint8_t *value);
```

9. Button APIs:

```
void Button_init(uint8_t buttonPort, uint8_t buttonPin);
```

```
void Button_read(uint8_t buttonPort, uint8_t buttonPin, uint8_t *value);
```

10. Ultrasonic sensor module APIs:

```
void ULTRASONIC_init(void);
```

```
uint16_t ULTRASONIC_read(Ultrasonic_type ultra);
```

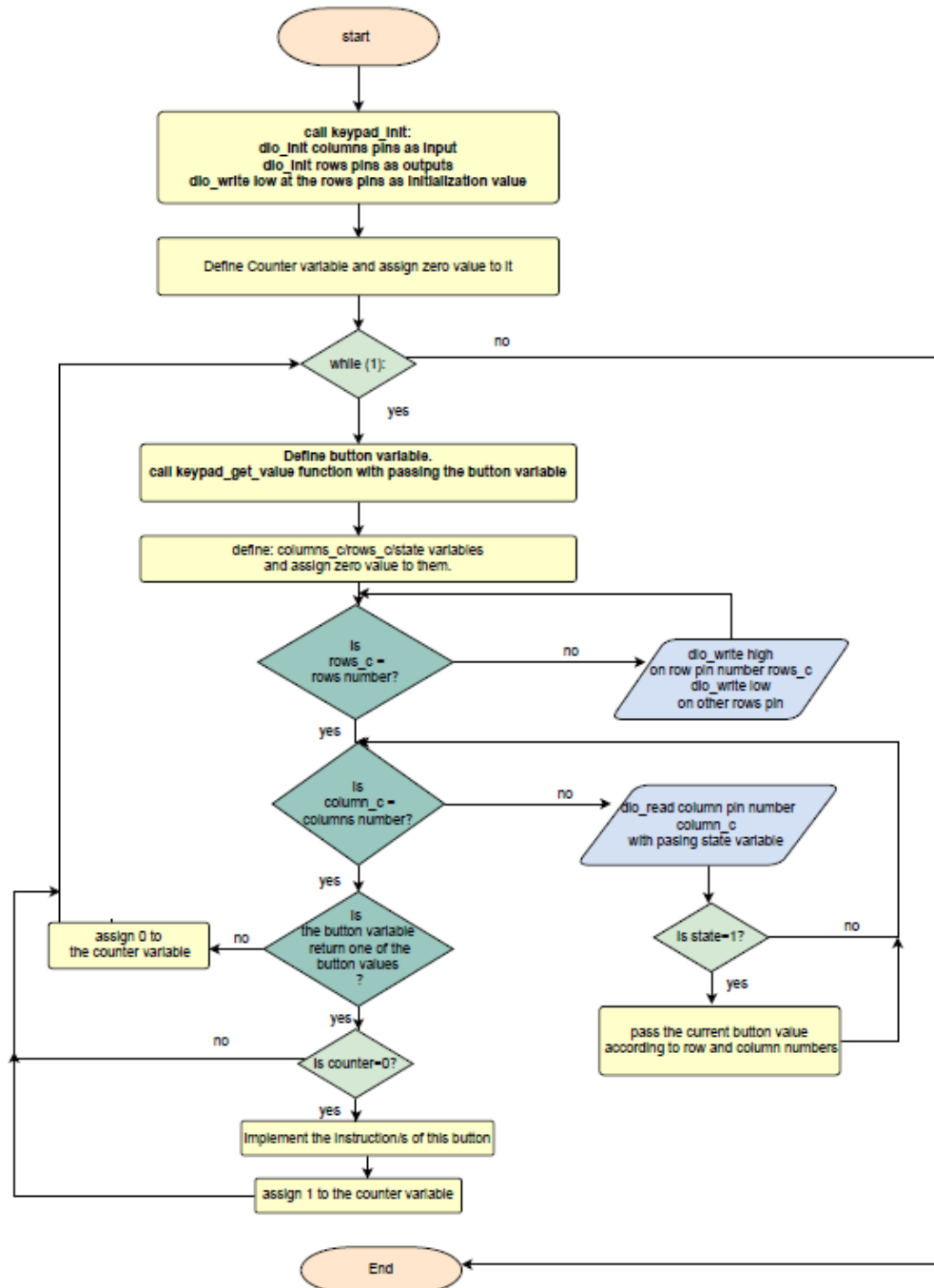
```
void ULTRASONIC_start(Ulmtrasonic_type ultra);
```

```
uint8_t ULTRASONIC_getReadNoBlock(uint16_t *distance_ptr);
```

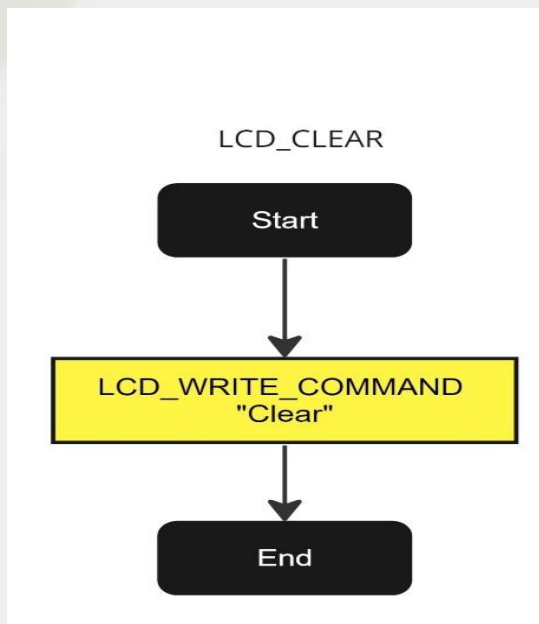
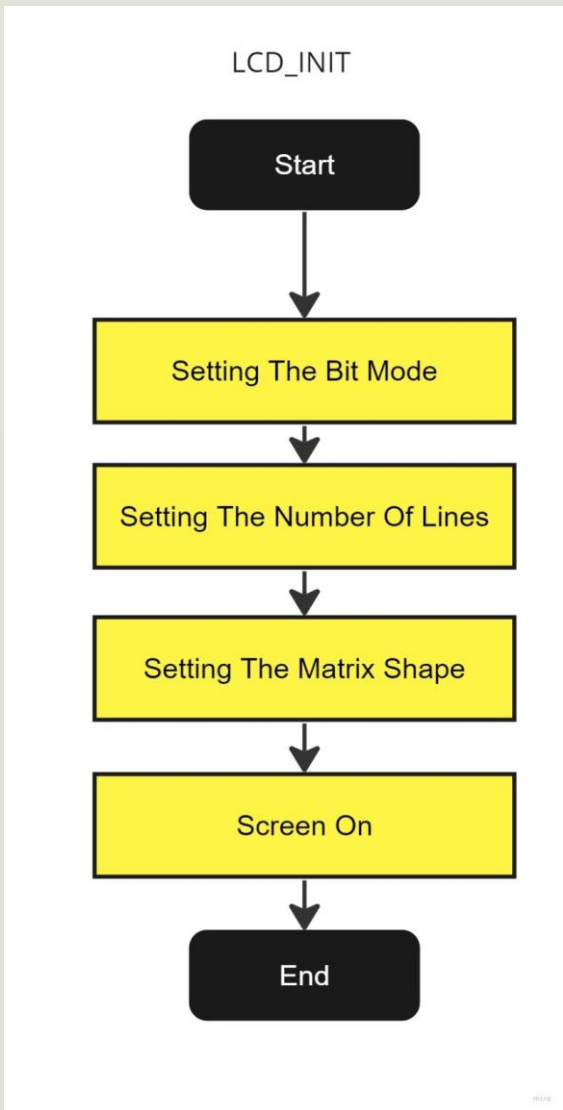
LOW LEVEL DESIGN

Flow charts for module's functions

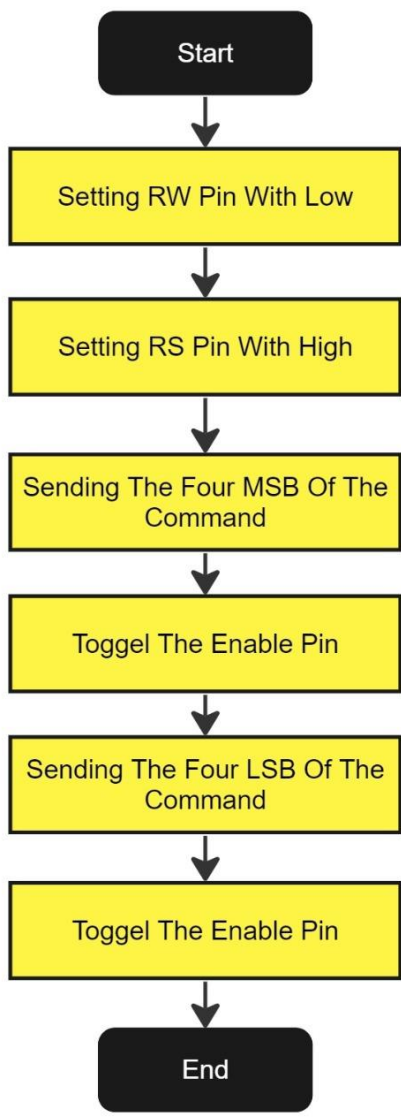
Keypad:



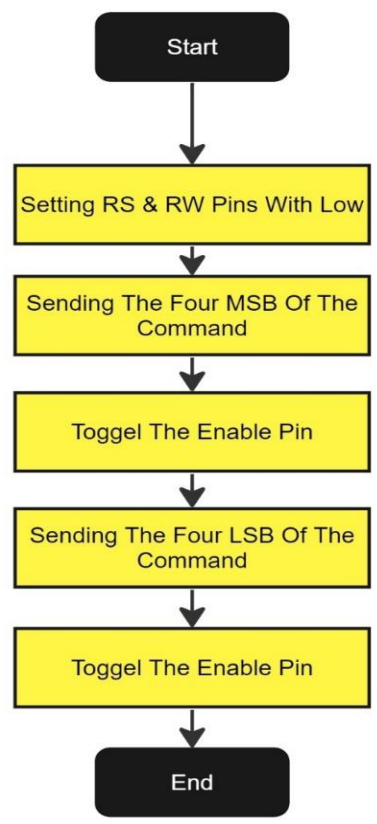
LCD:



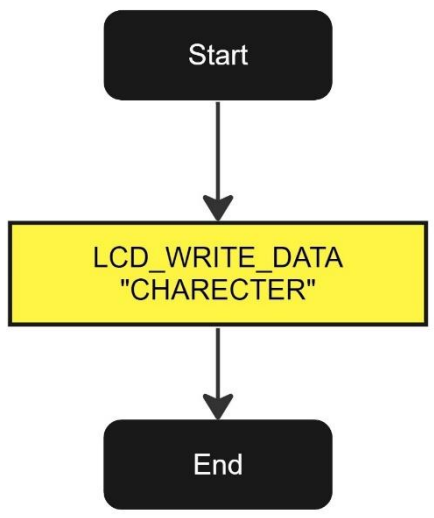
LCD_WRITE_DATA

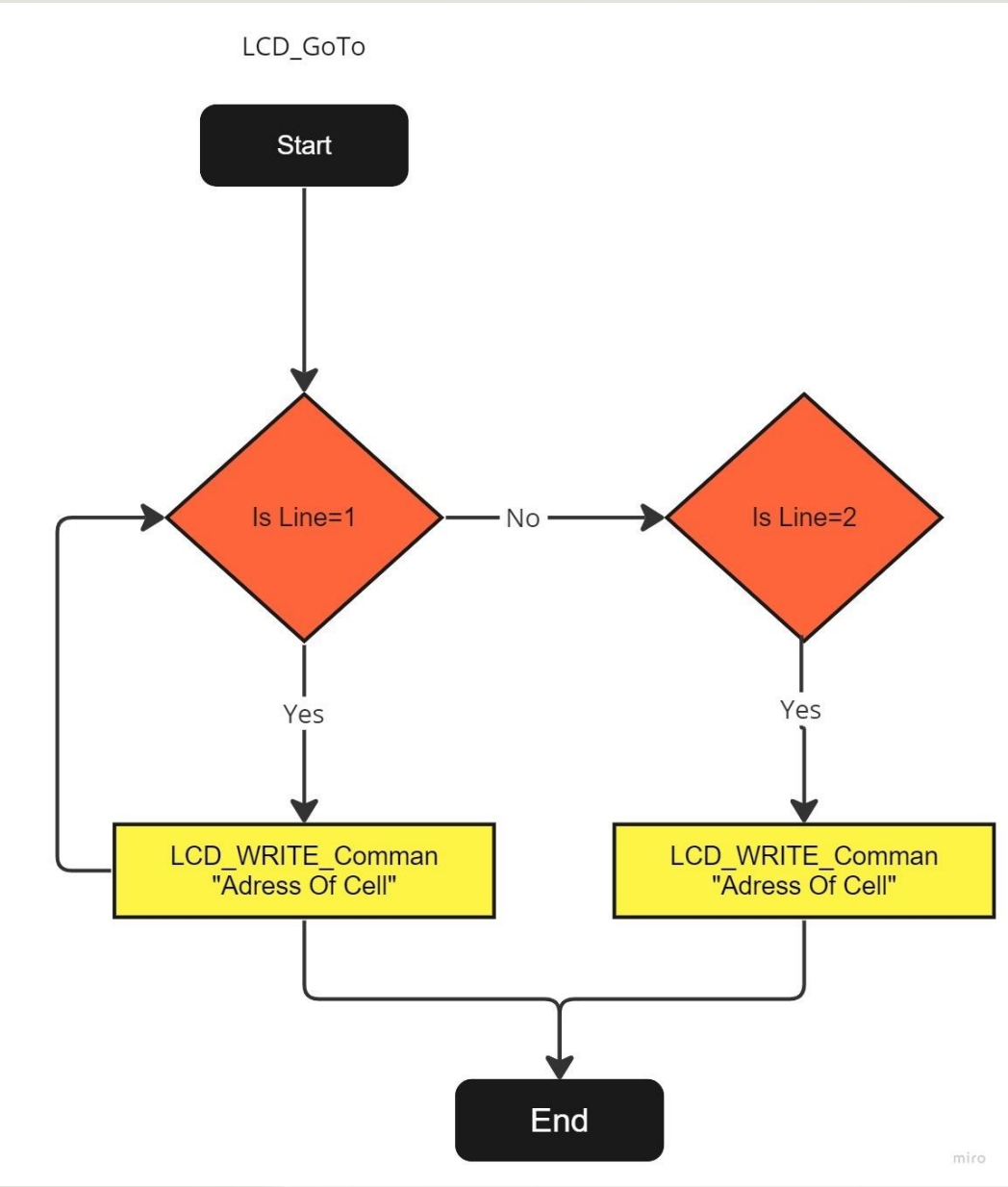


LCD_WRITE_COMMAND

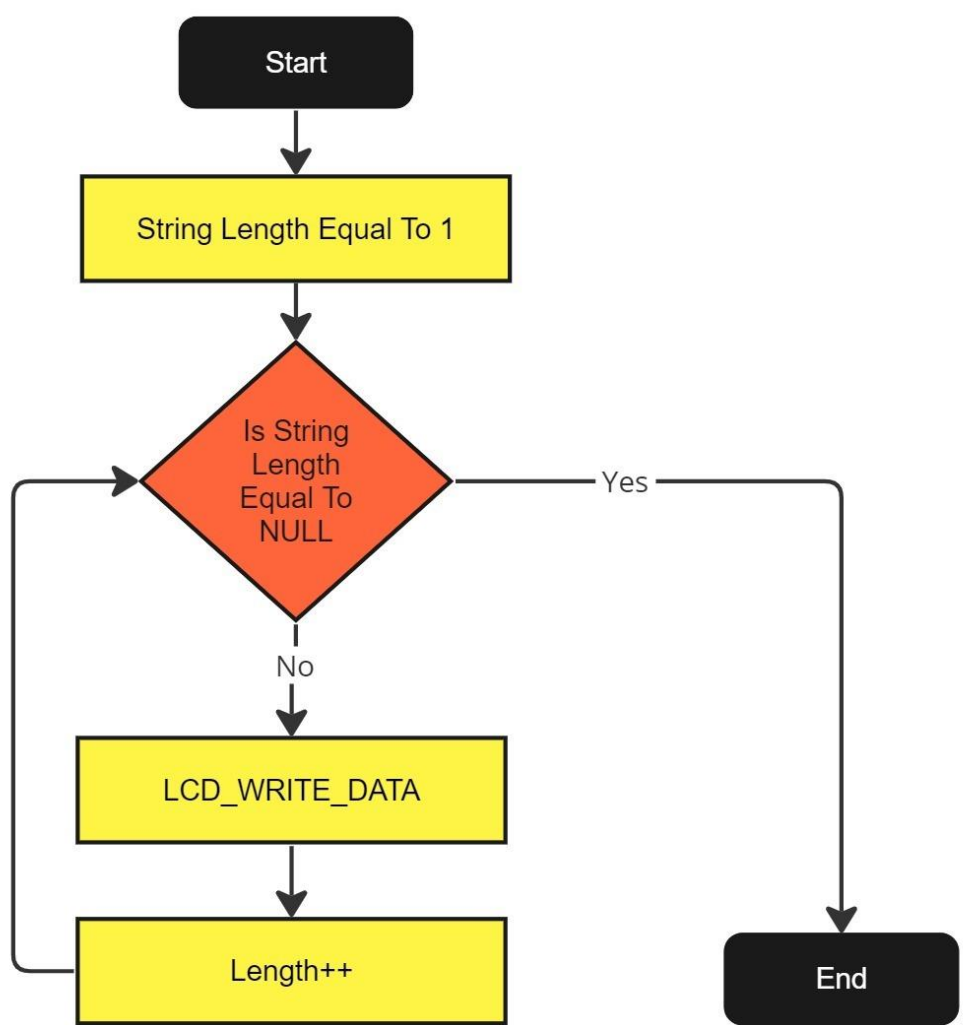


LCD_Write_Charecter

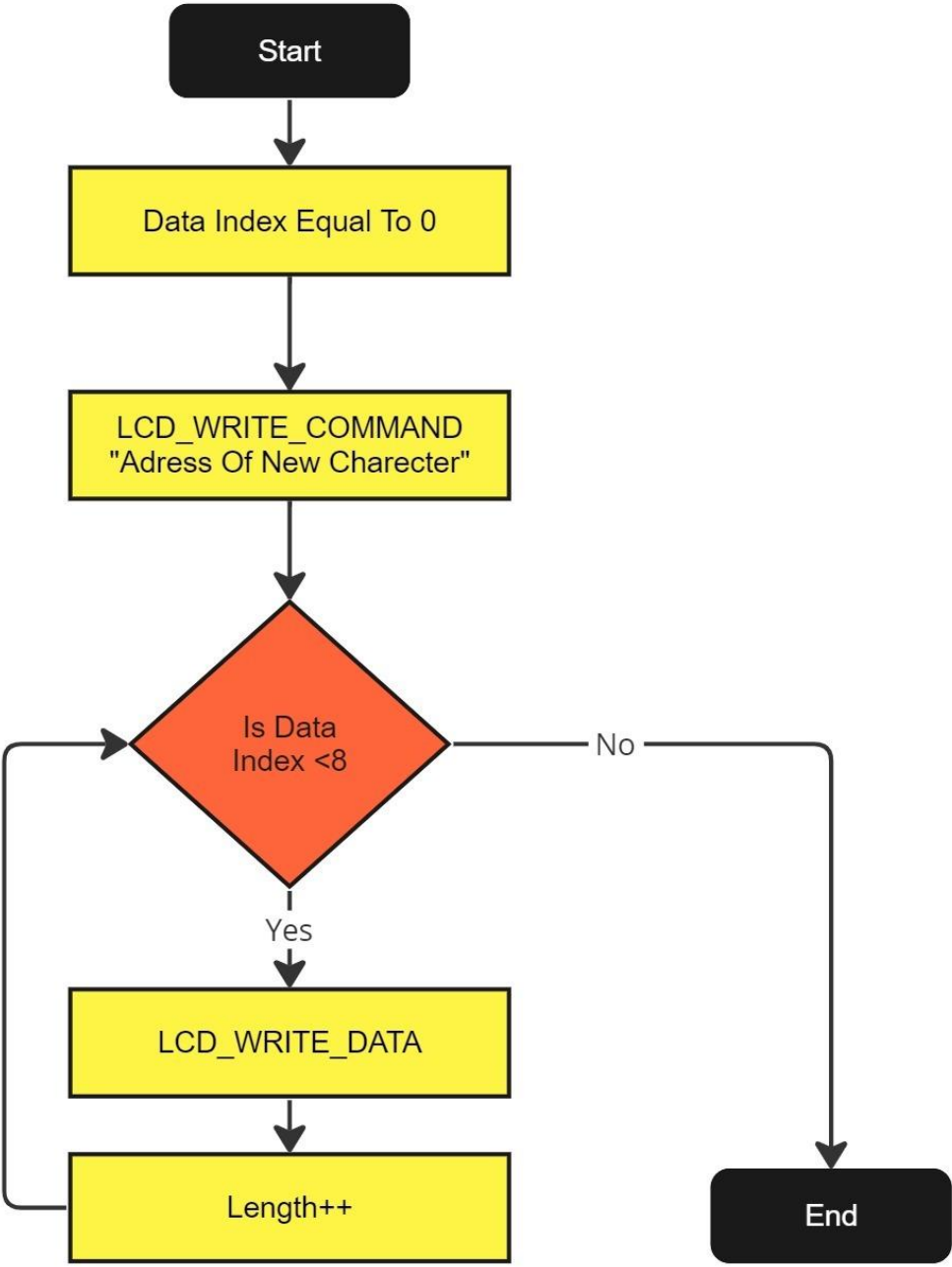




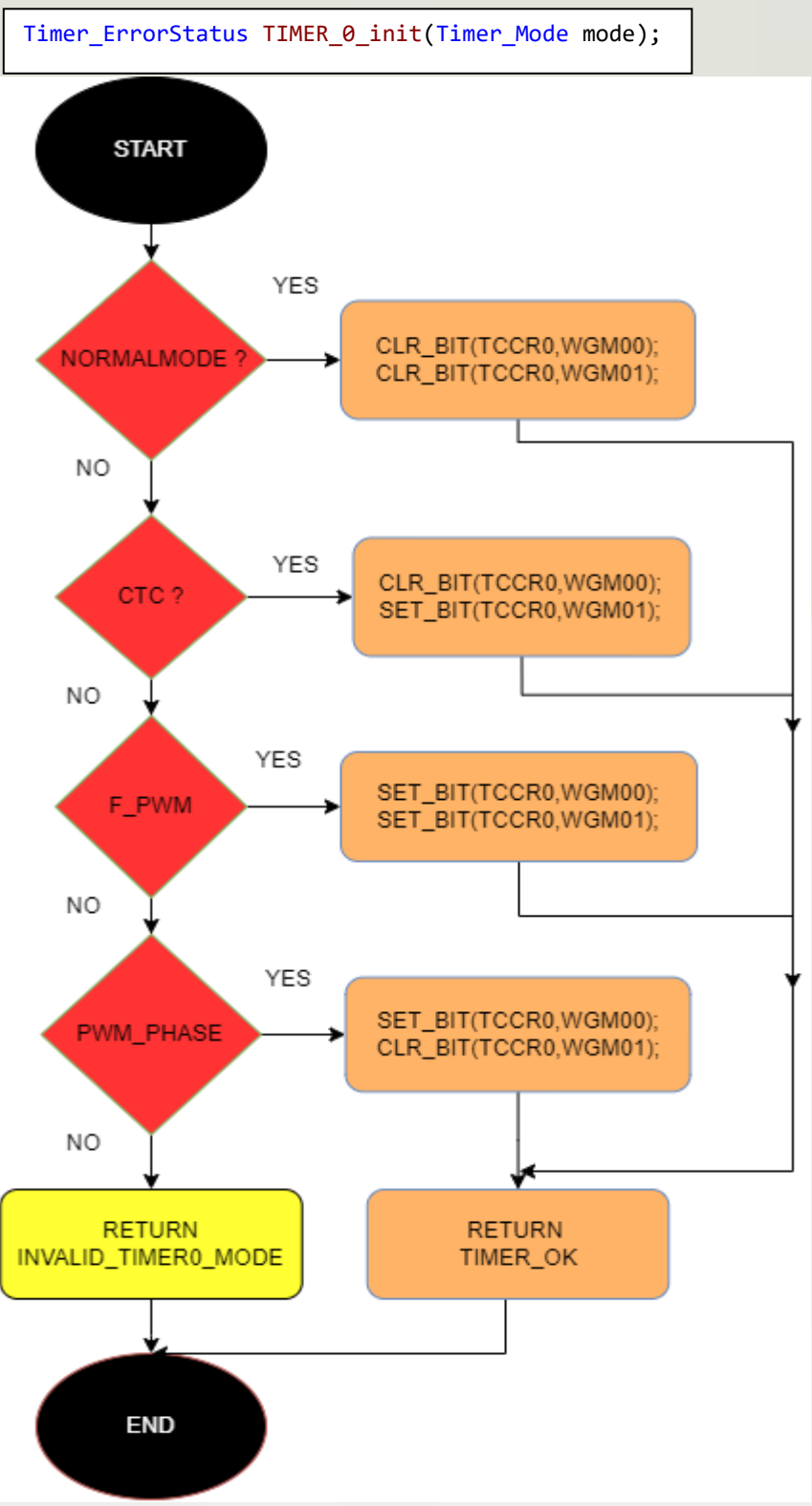
LCD_Write_String



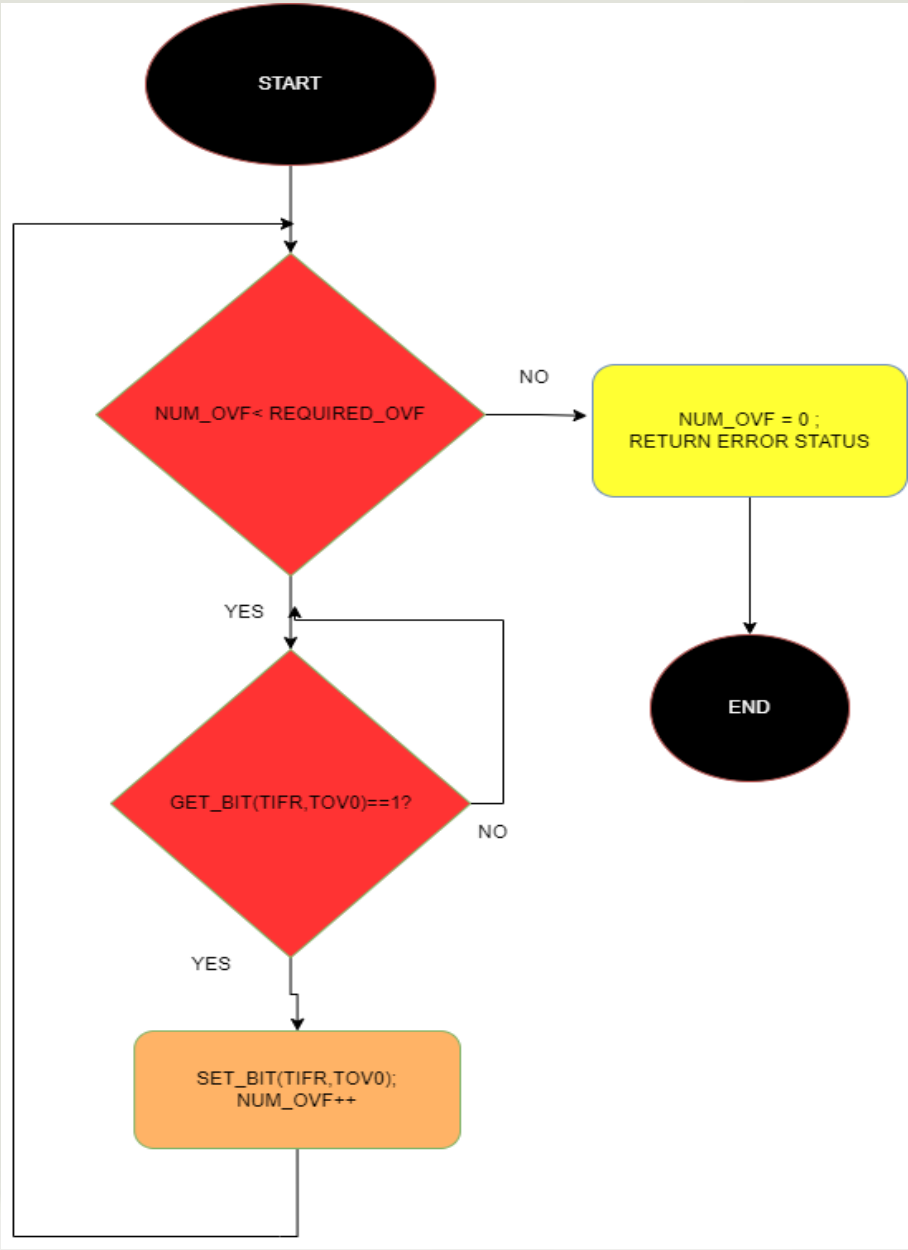
LCD_Create_Charecter



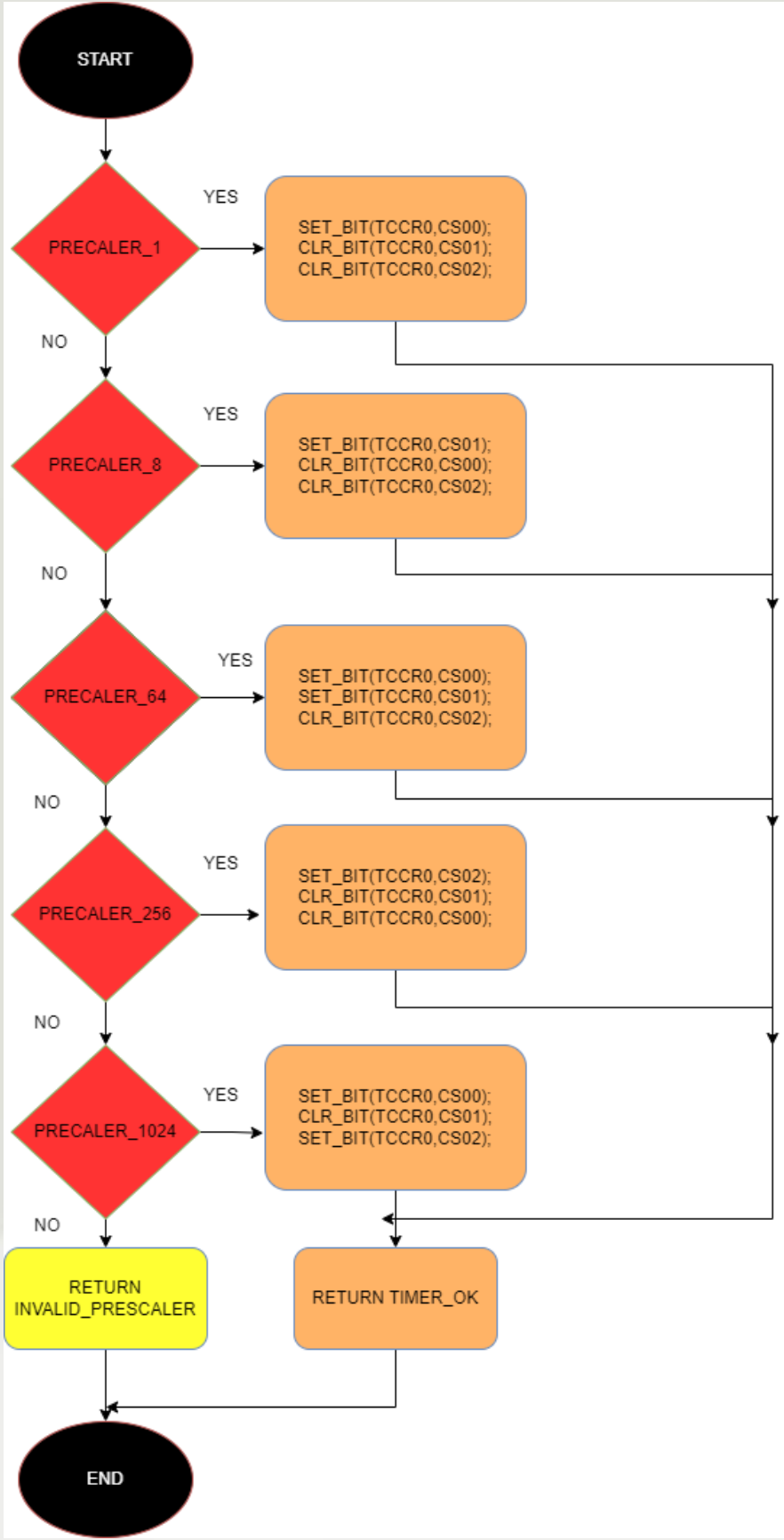
Timer:



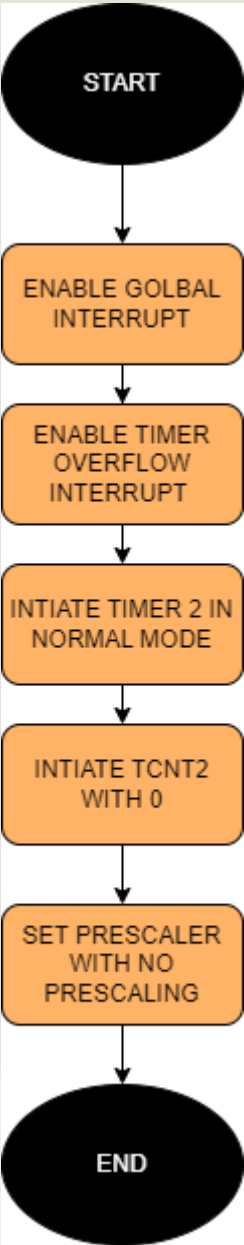
```
Timer_ErrorStatus TIMER_0_OvfNum(double overflow);
```



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

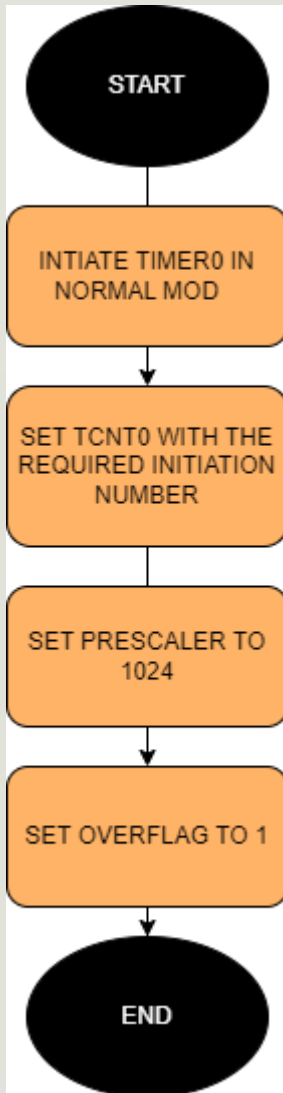


TIMER 2 WITH INTERRUPT

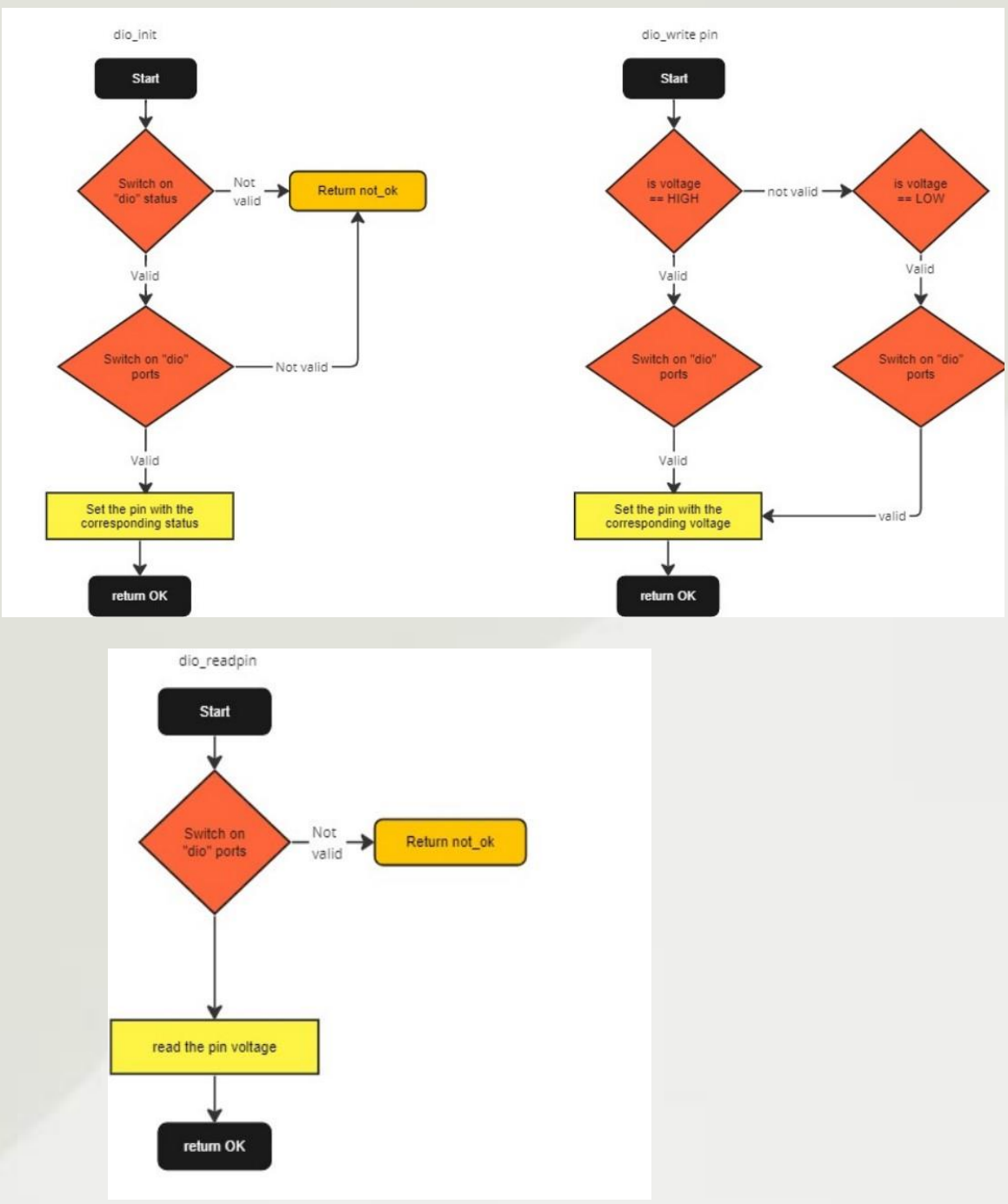


PWM:

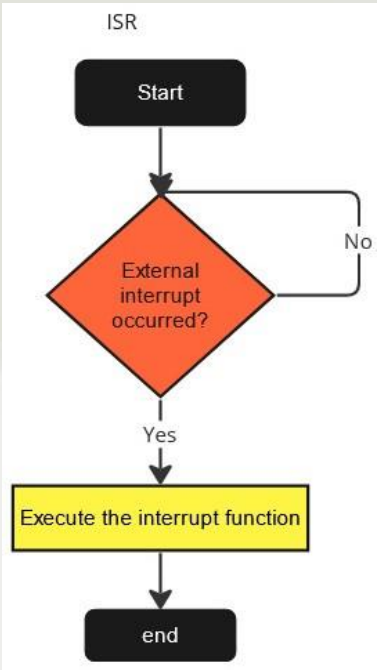
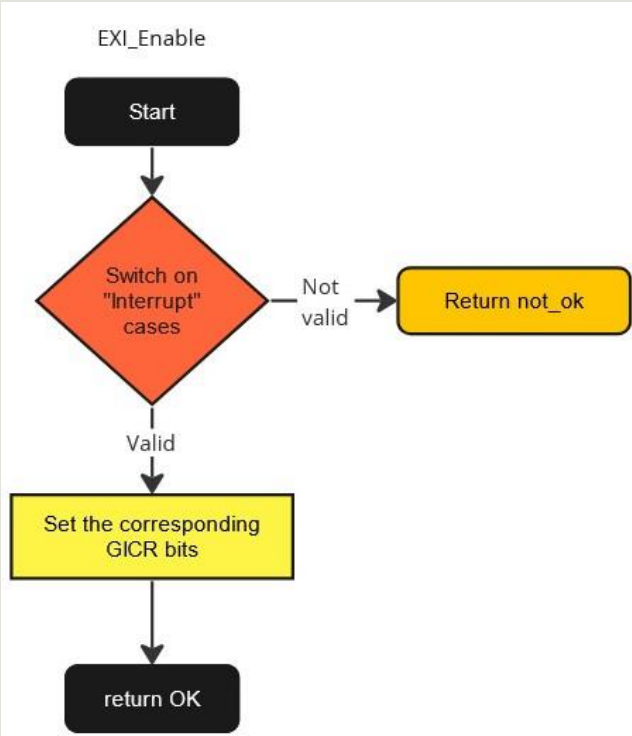
```
void TIMER_0_pwm(float intial);
```

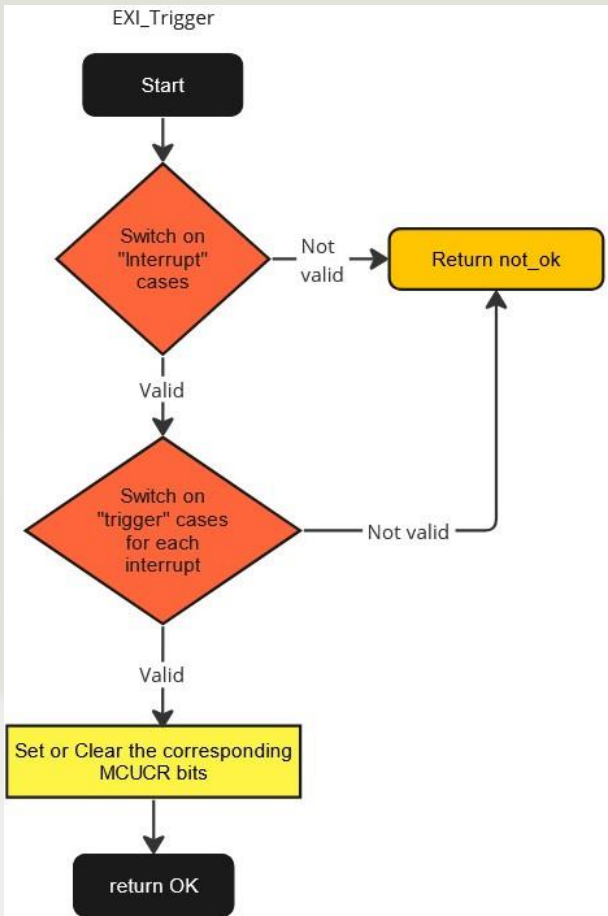
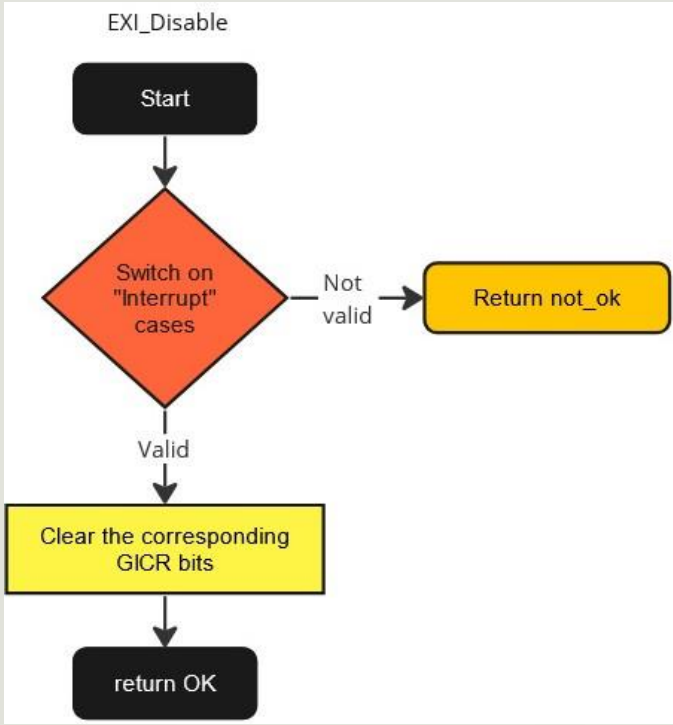


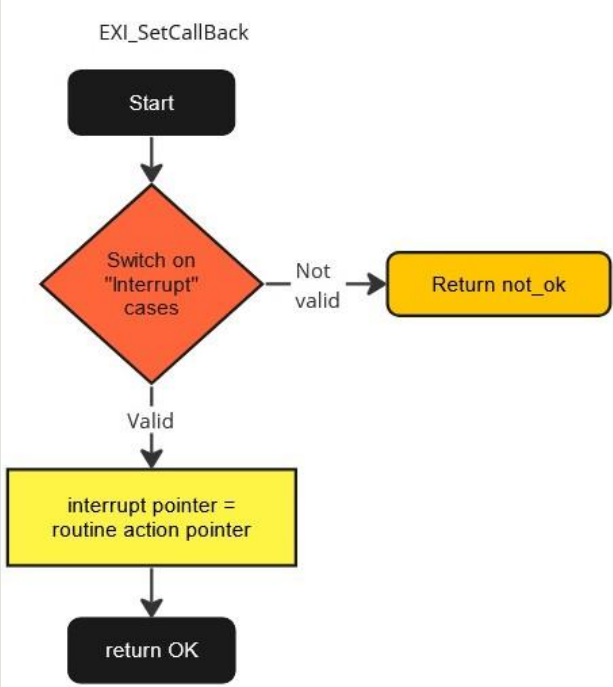
DIO:



Interrupt:







Dc motor:

