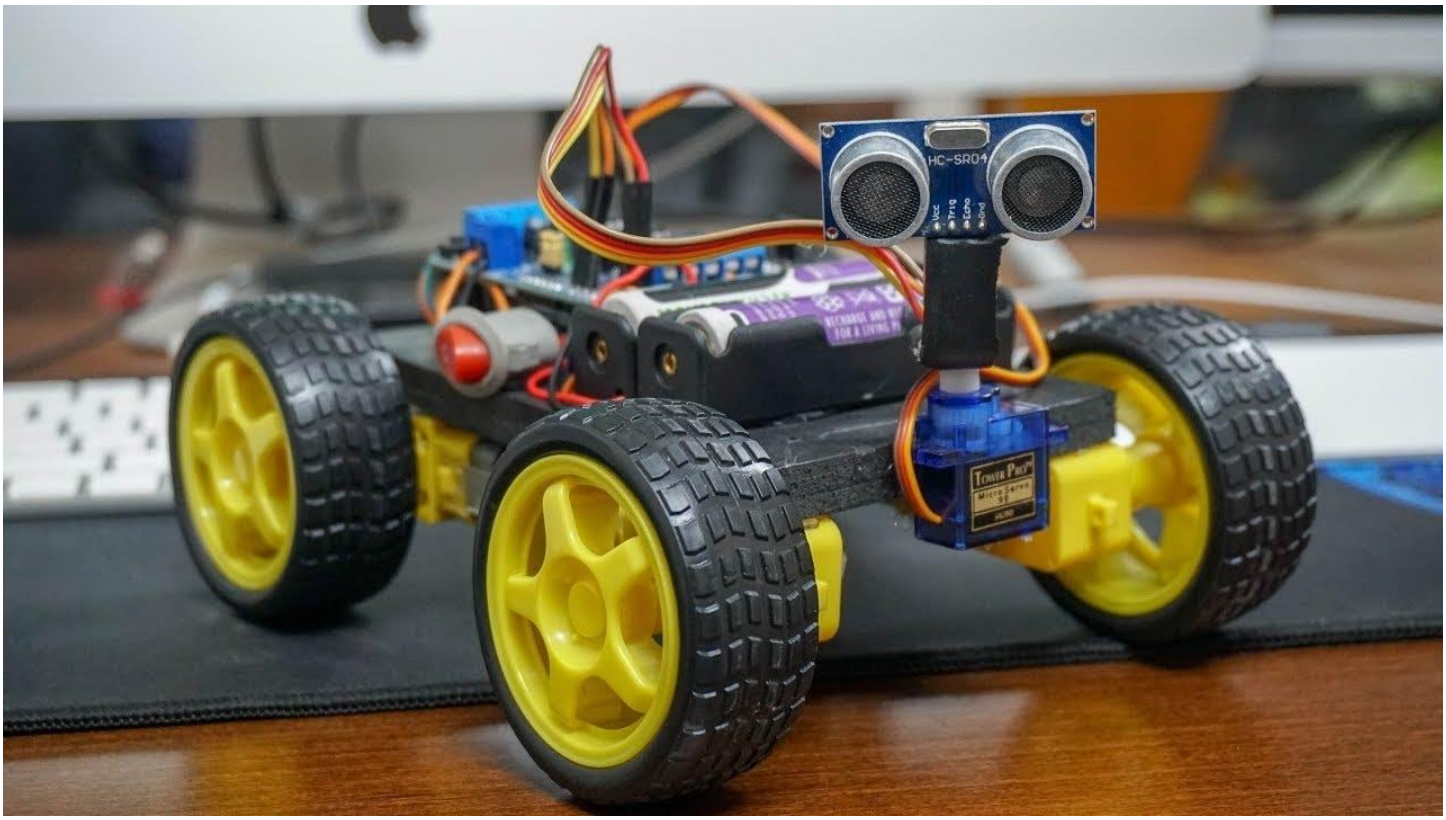
Obstacle Avoidance Robot V1.0 Design



Anas Mahmoud	Team 2

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)

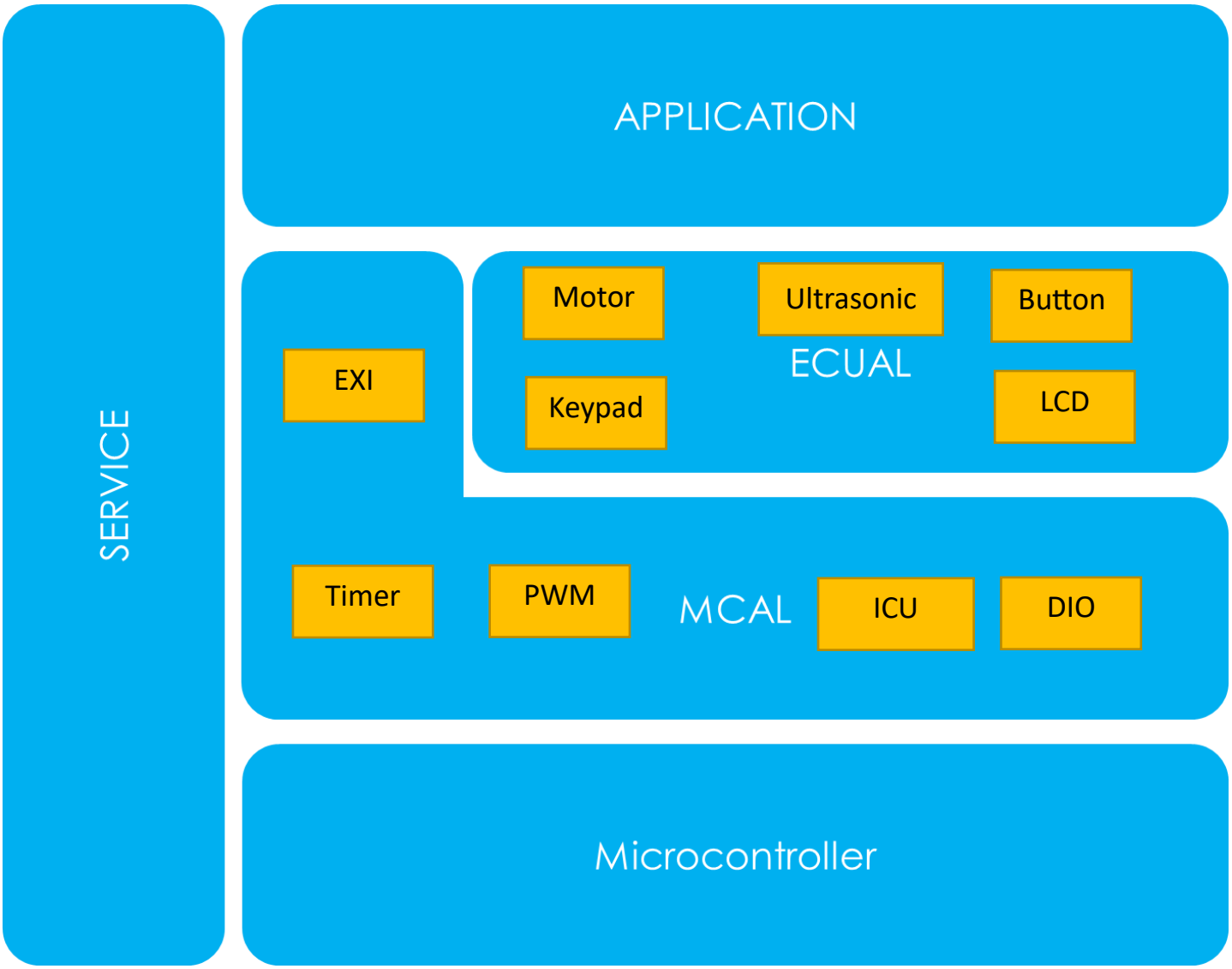
1. If the robot rotated for 360 degrees without finding any distance greater than 20 it will stop

2. LCD data will be updated.

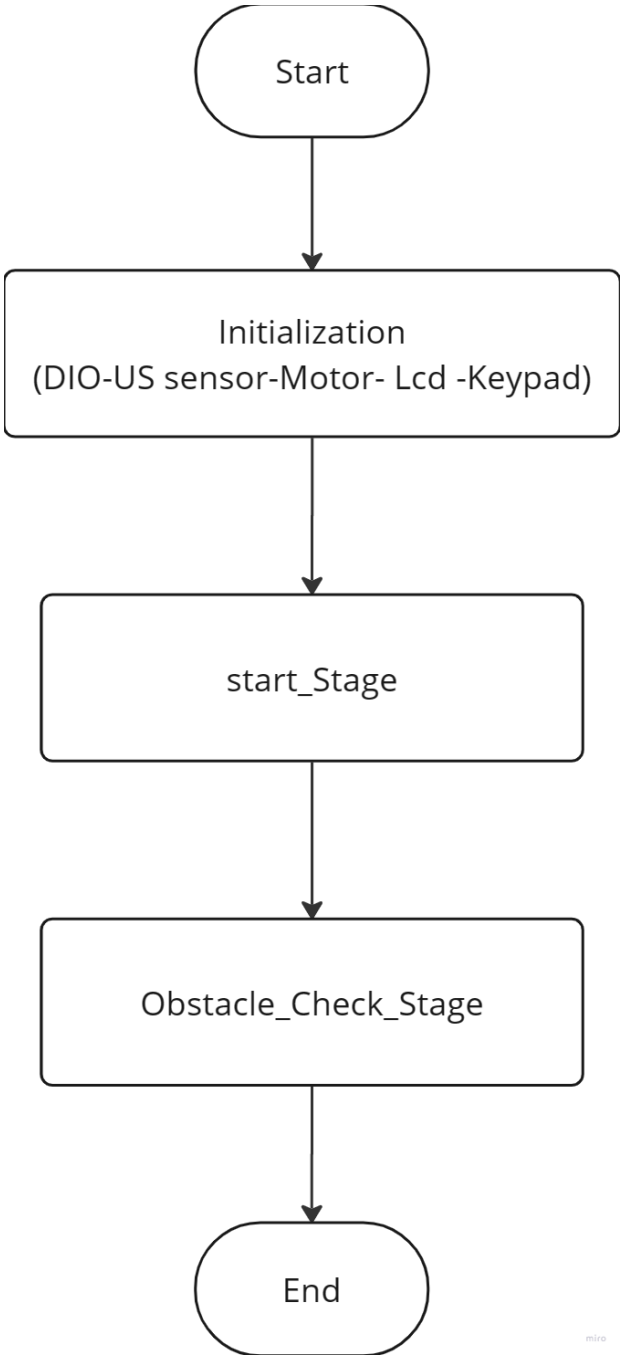
3. The robot will frequently (each 3 seconds) check if any of the obstacles was removed or not and move in the direction of the furthest object

2-High Level Design

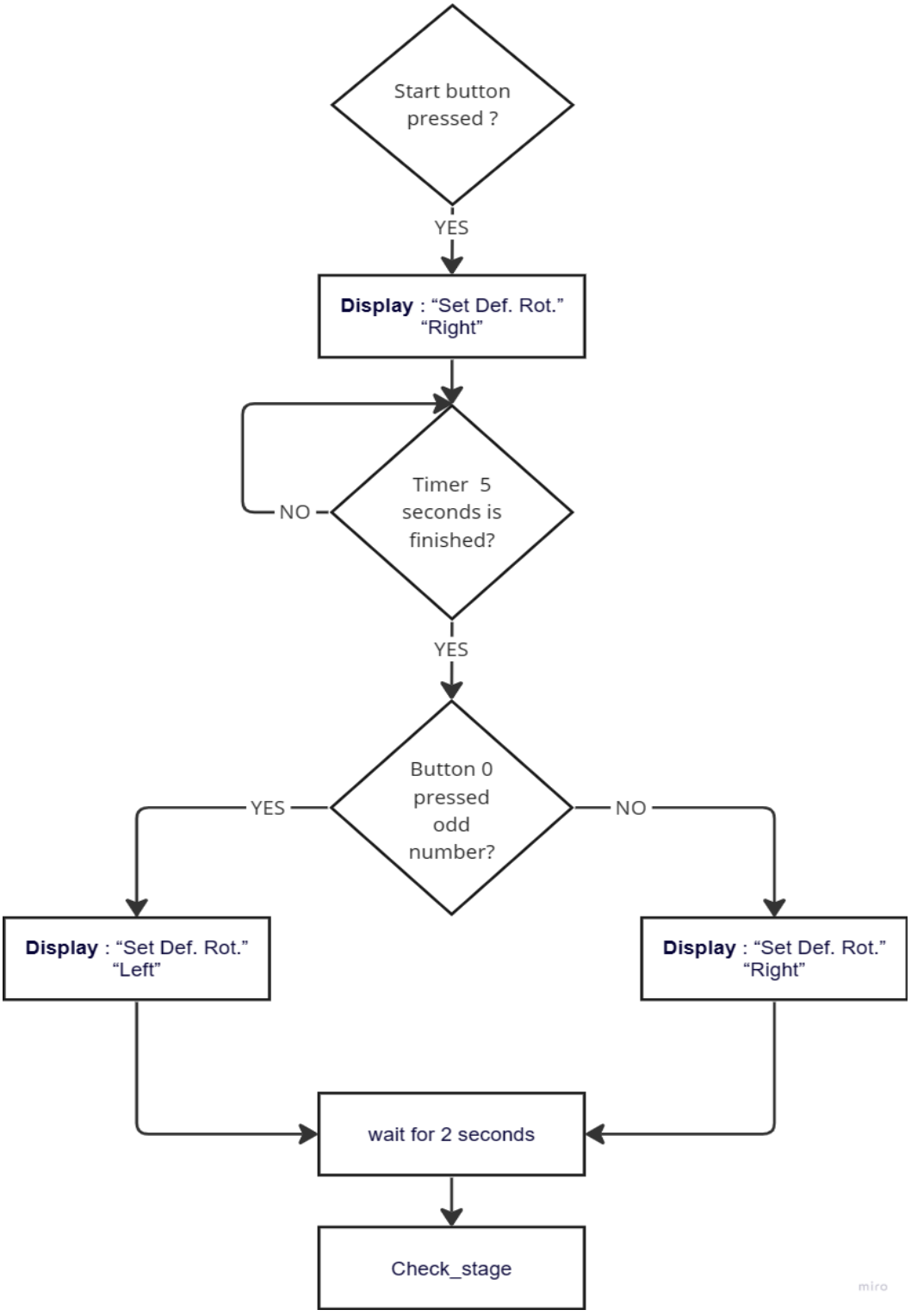
2.1 Layered Architecture



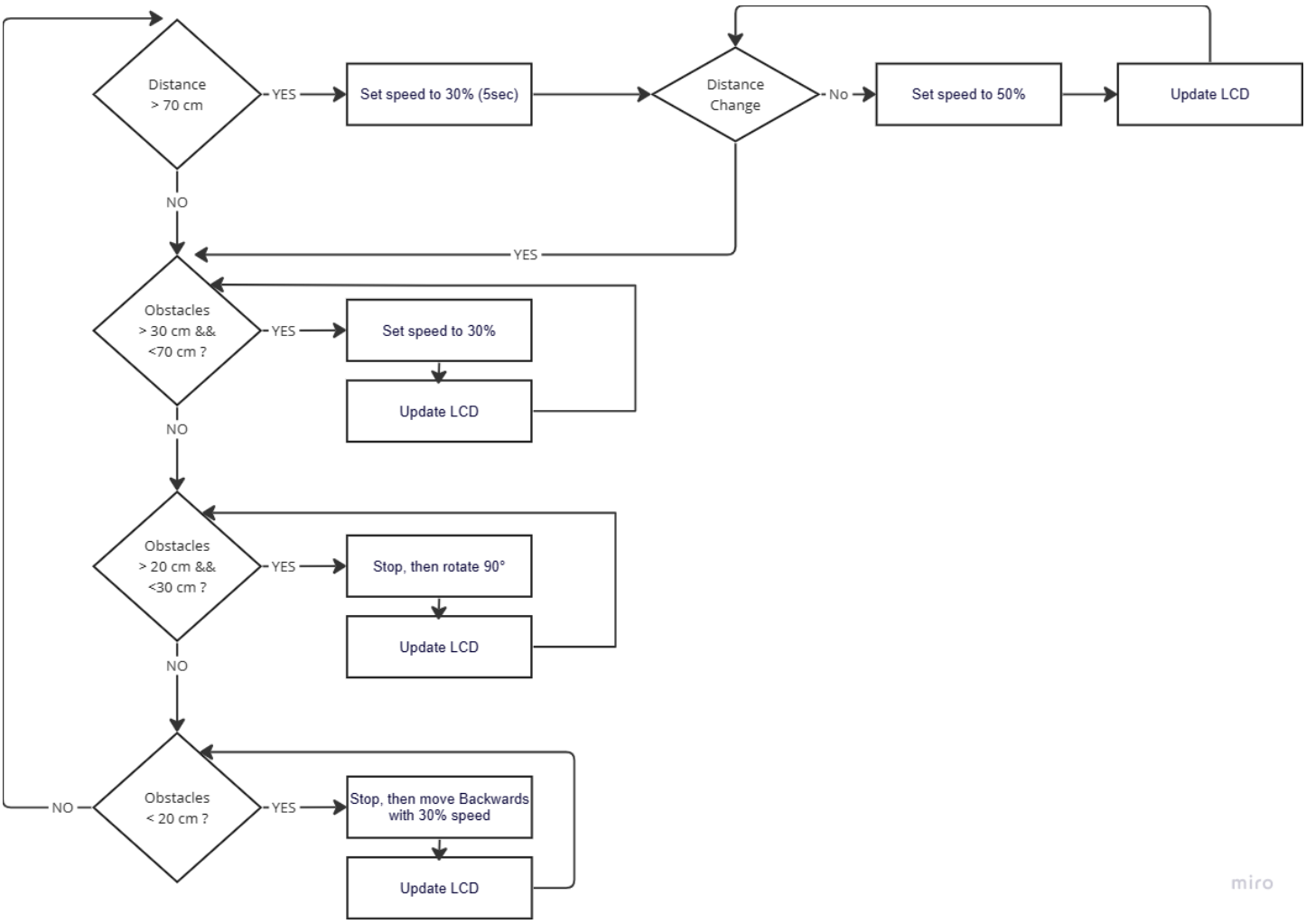
2.2 System Flow Chart



Start_Stage


Start button pressed ?
YES
Display : “Set Def. Rot.” “Right”
Timer 5 seconds is finished?
NO
YES
Button 0 pressed odd number?
YES
NO
Display : “Set Def. Rot.” “Left”
Display : “Set Def. Rot.” “Right”
wait for 2 seconds
Check_stage
miro

Obstacle_Check_Stage



2.3 Drivers Descriptions

2.3.1 DIO Driver

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

Location: MCAL

Function: used to set a time delay

2.3.3 PWM Driver

Location: MCAL

Function: used to control motor speed

2.3.4 EXI Driver

Location: MCAL

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

2.3.5 ICU Driver

Location: MCAL

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

2.3.6 Keypad Driver

Location: HAL

Function: used to initialize the keypad, get pressed key

2.3.7 Button Driver

Location: HAL

Function: used to initialize the button, check the button status pressed or not

2.3.8 LCD Driver

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

Location: HAL

Function: used to detect the distance between car & obstacle

2.3.10 Application Driver

Location: App

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

2.4 Modules API's

2.4.1 DIO Module

```
en_dioError_t  DIO_initpin  (DIO_Pin_type pin,DIO_PinStatus_type status);
en_dioError_t  DIO_writepin (DIO_Pin_type pin,DIO_PinVoltage_type volt);
en_dioError_t  DIO_readpin  (DIO_Pin_type pin,DIO_PinVoltage_type *volt);
en_dioError_t  DIO_togglepin(DIO_Pin_type pin);
en_dioError_t  DIO_WritePort (DIO_Port_type port,u8 value);
```

2.4.2 Timer Module

```
Void TMR0_init(void);
TMR0_delay_error TMR0_delayms(uint32_t u32_a_delayms);
TMR0_delay_error TMR0_delaymicos(uint32_t u32_a_delaymicros);
```

2.4.3 PWM Module

```
Void PWM_init(void);
Void Start-Signal(u16 freq , u8 duty_cycle);
Void Stop_Signal();
```

2.4.4 Keypad Module

```
KEYPAD_initError KEYPAD_init(void) ;
KEYPAD_readError KEYPAD_getpressedkey(u8 *value) ;
```

2.4.5 Button Module

```
Button_State Is_pressed( u8 BUTTON_PIN , u8 *value);
```

2.4.6 LCD Module

LCD_init_error LCD_8_bit_init(void);

LCD_sendCommand_error LCD_8_bit_sendCommand(uint8_t u8_a_command);

LCD_sendChar_error LCD_8_bit_sendChar(uint8_t u8_a_char);

LCD_init_error LCD_4_bit_init(void);

LCD_sendCommand_error LCD_4_bit_sendCommand(uint8_t u8_a_command);

LCD_sendChar_error LCD_4_bit_sendChar(uint8_t u8_a_char);

LCD_sendString_error LCD_sendString(uint8_t *u8_a_string);

void LCD_goTo(uint8_t u8_a_row,uint8_t u8_a_column);

void LCD_createCustomCharacter(uint8_t *u8_a_bitMap,uint8_t u8_a_location);

LCD_init_error LCD_init(void);

LCD_sendCommand_error LCD_sendCommand(uint8_t u8_a_command);

LCD_sendChar_error LCD_sendFloat(float f_a_number);

LCD_sendChar_error LCD_sendInteger(uint16_t u16_a_number);

2.4.7 Ultrasonic Module

void US_Init(void);

void US_Trigger(void);

2.4.8 App Module

Void App_init();

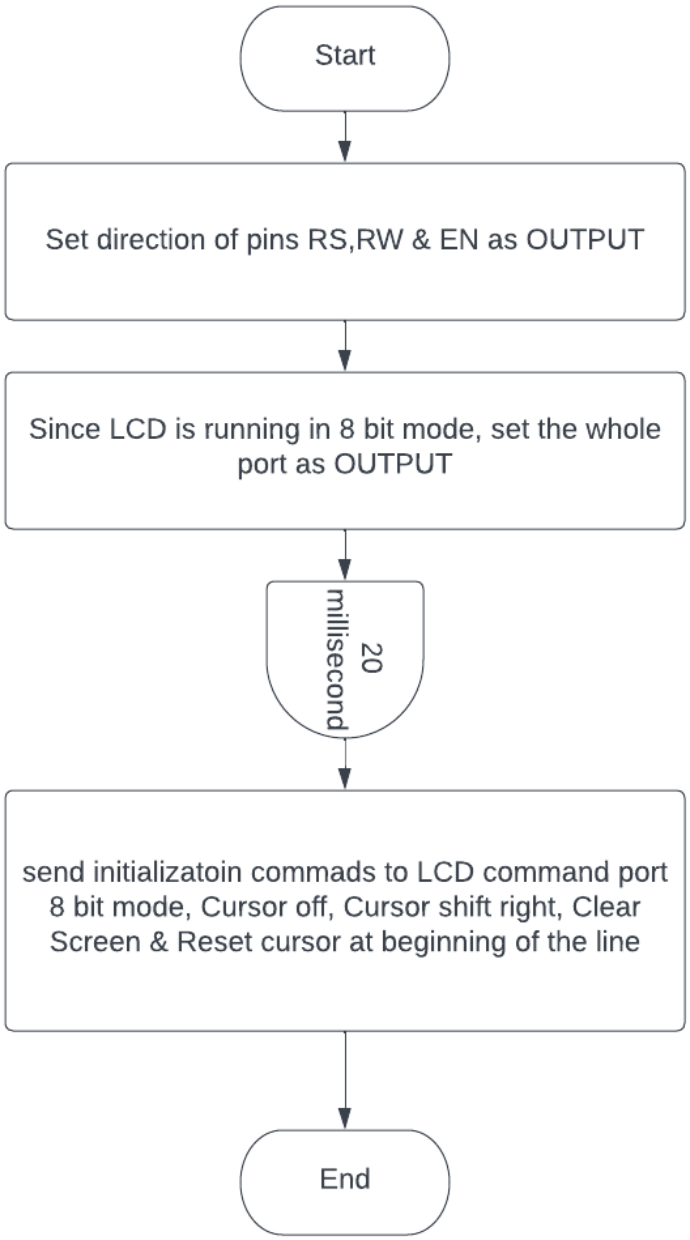
Void App_start();

3-Low Level Design

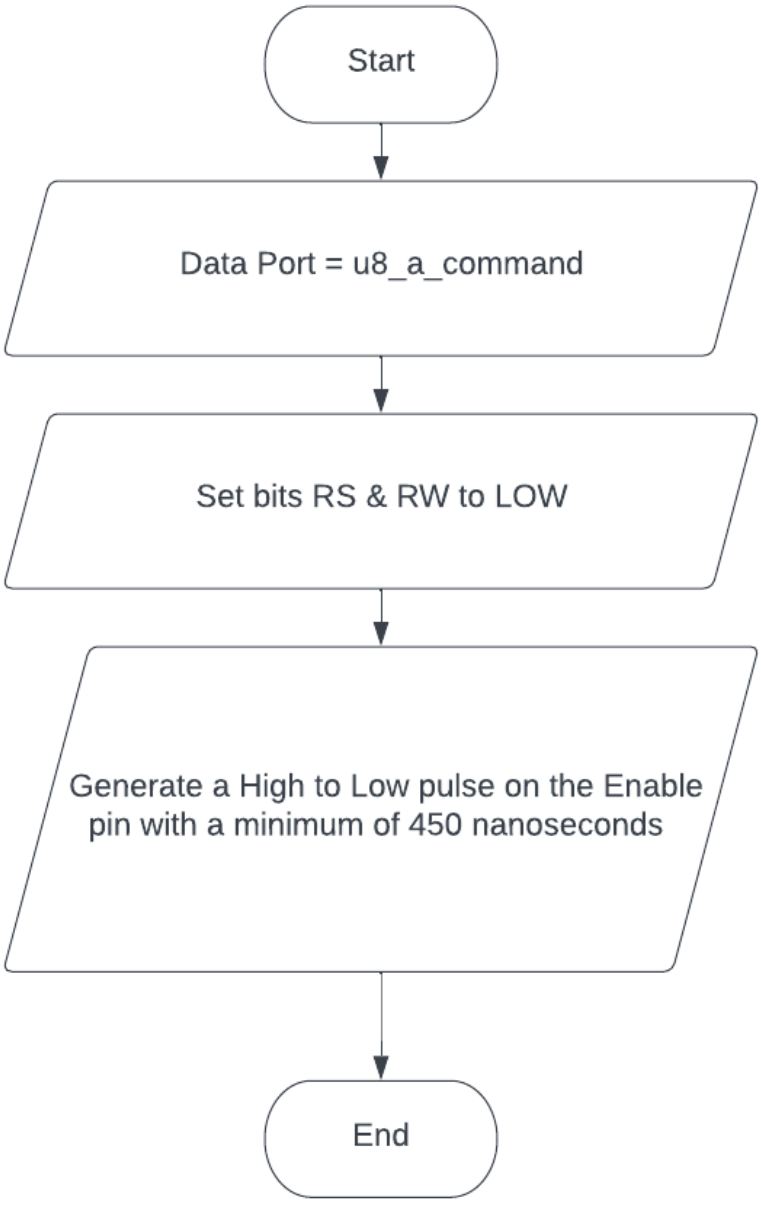
3.1 APIs Flow Chart

LCD APIs

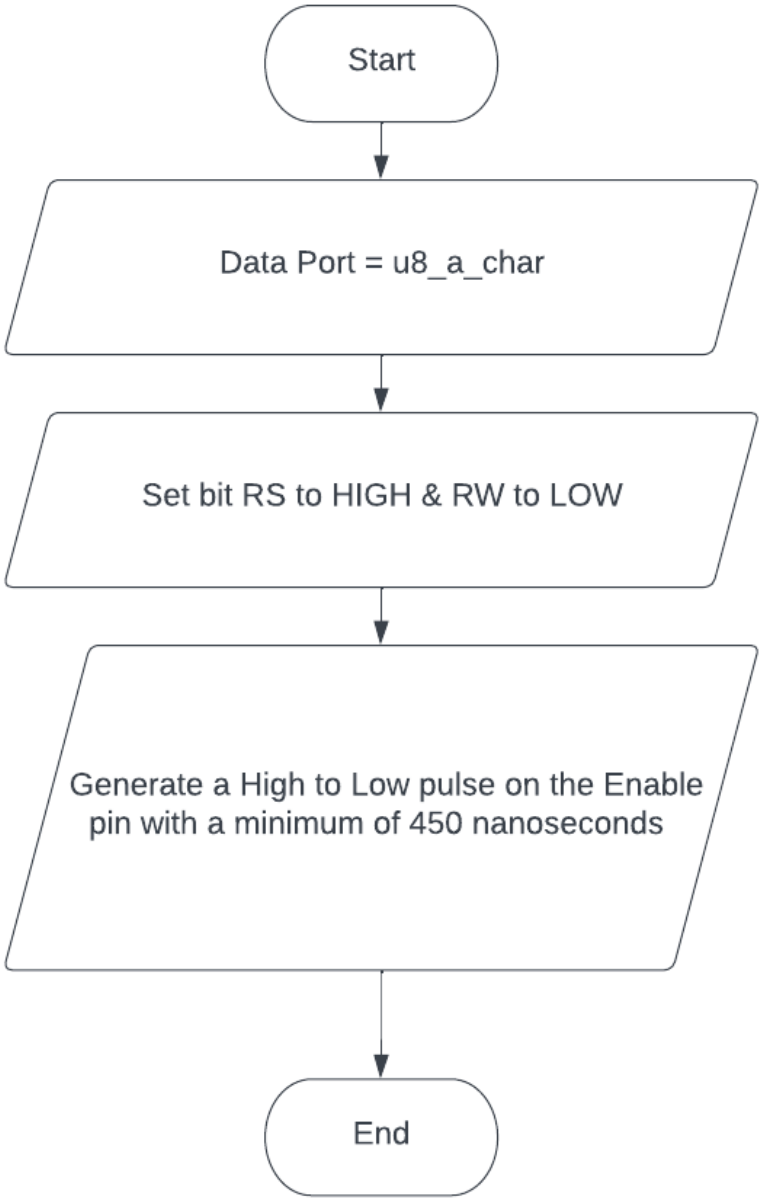
```
LCD_init_error LCD_8_bit_init(void)
```



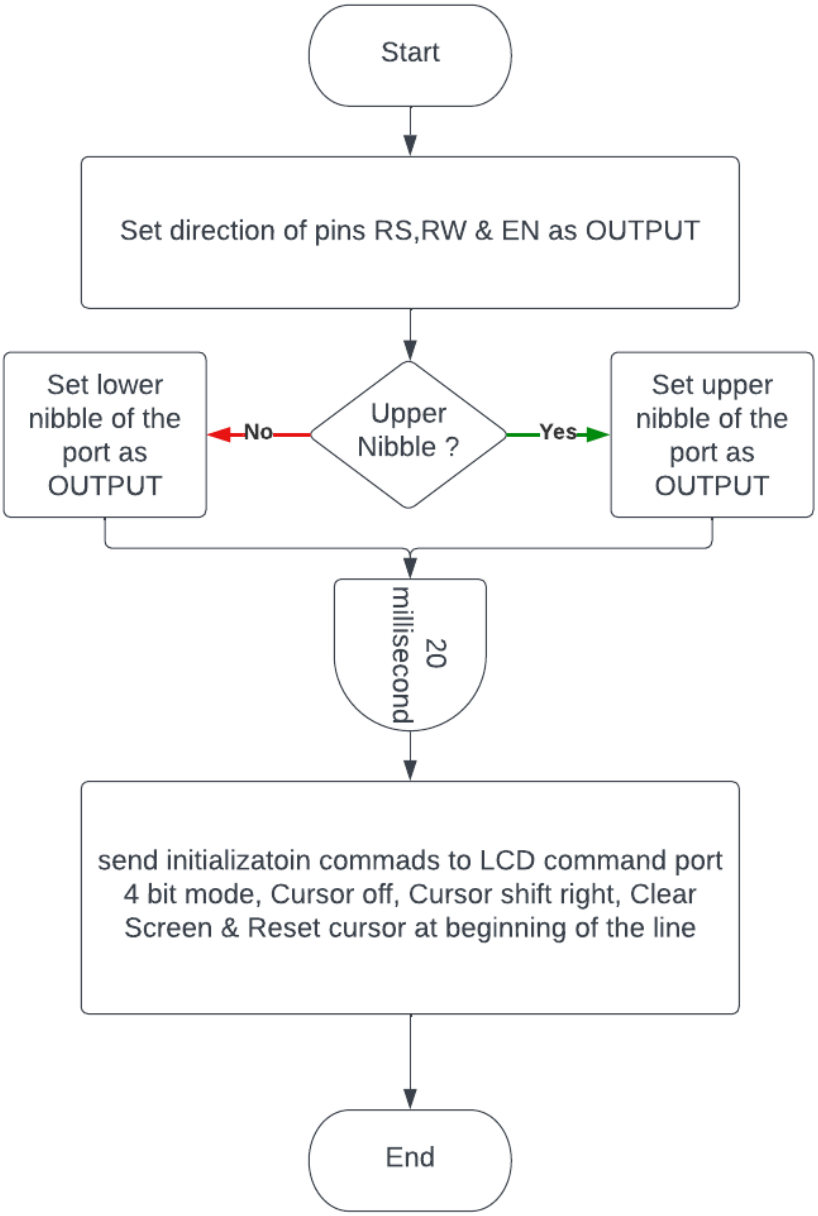
```
LCD_sendCommand_error LCD_8_bit_sendCommand(uint8_t u8_a_command);
```



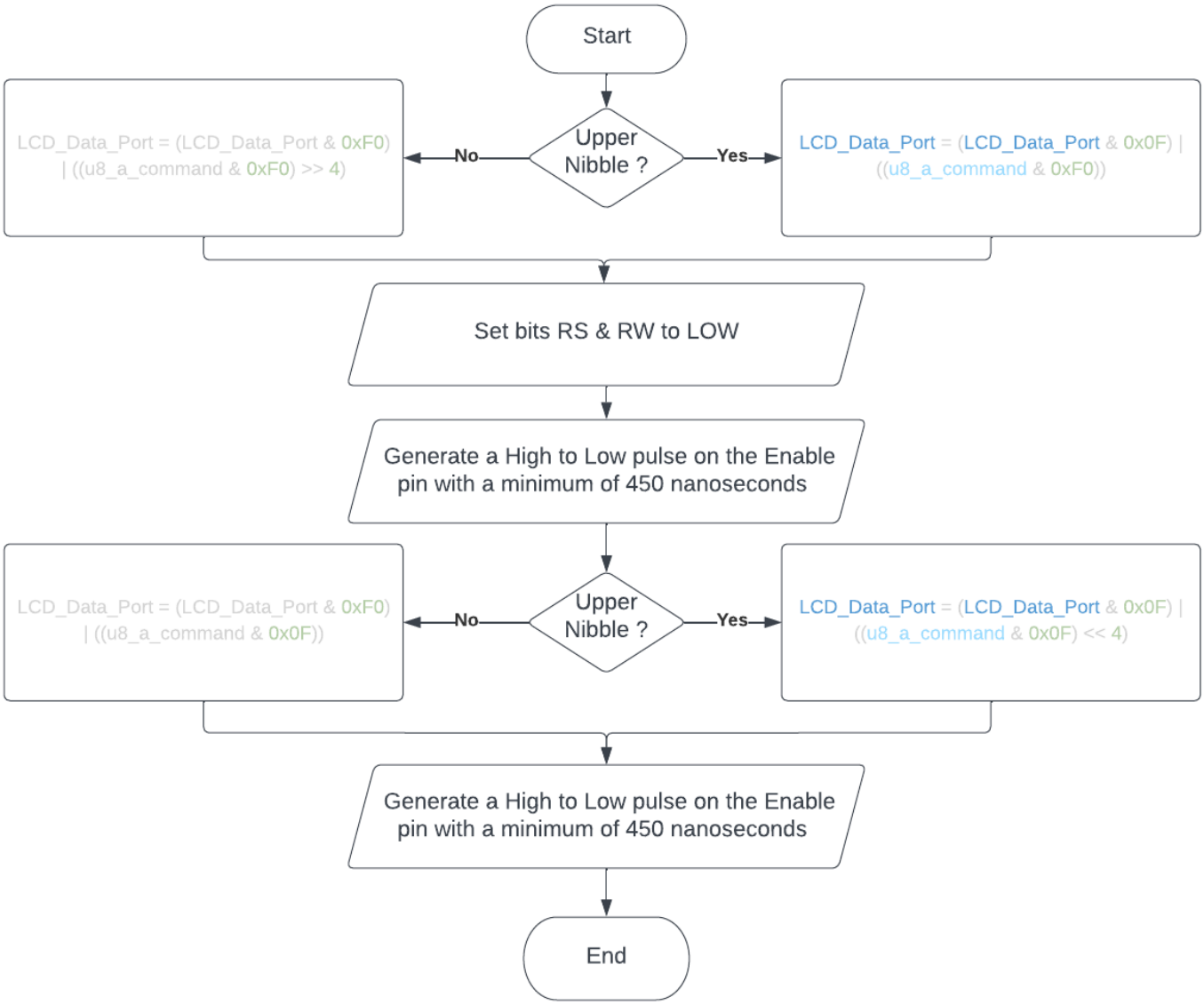
```
LCD_sendChar_error LCD_8_bit_sendChar(uint8_t u8_a_char);
```



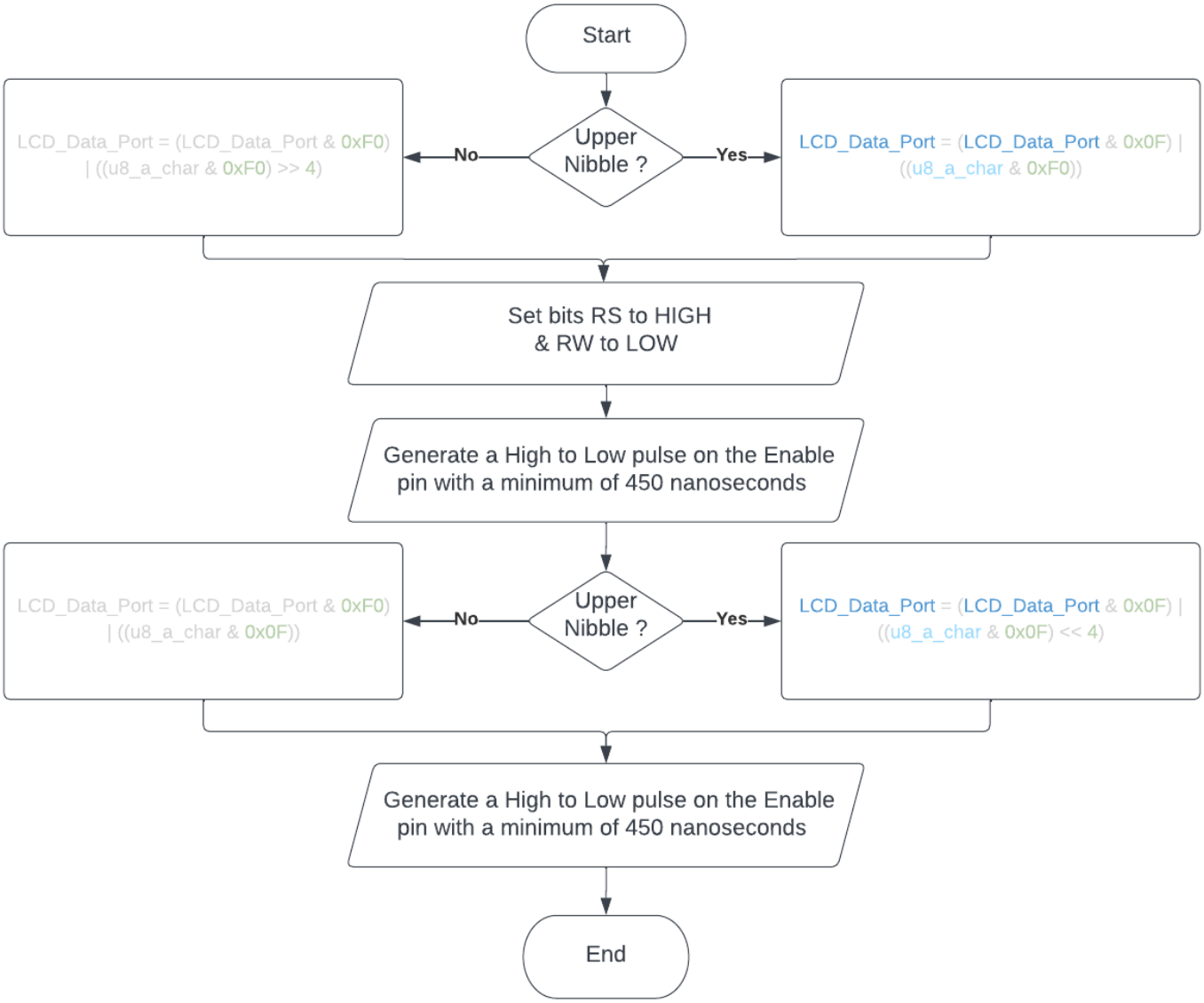
```
LCD_init_error LCD_4_bit_init(void);
```



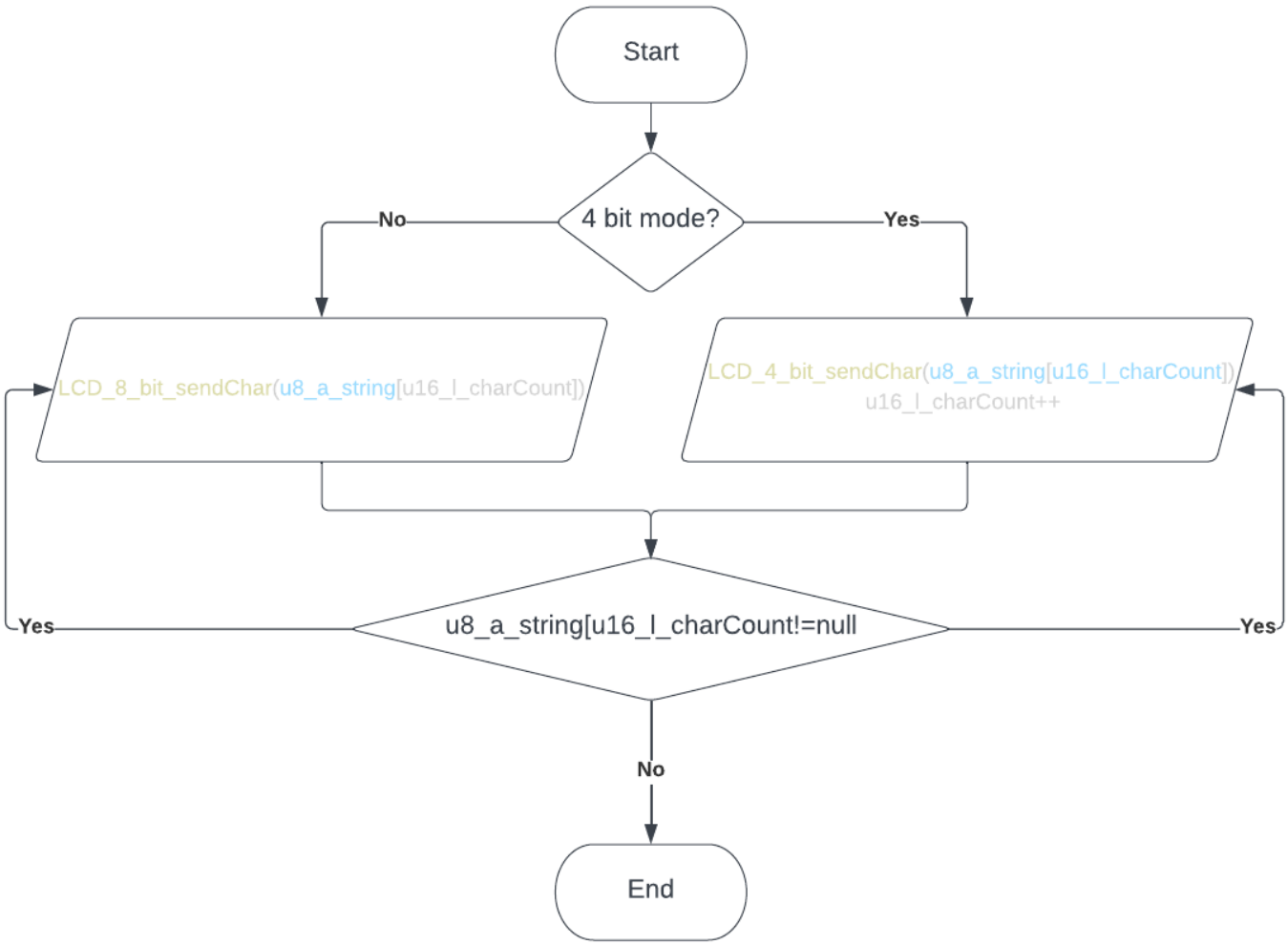
```
LCD_sendCommand_error LCD_4_bit_sendCommand(uint8_t u8_a_command);
```



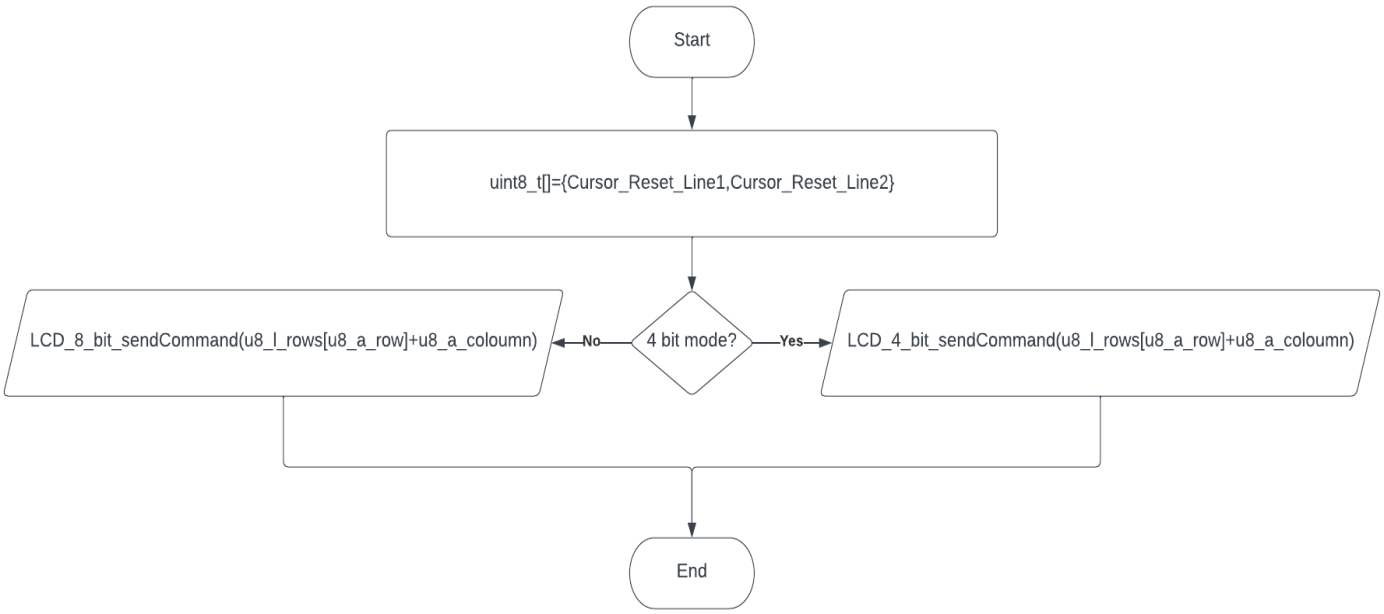
```
LCD_sendChar_error LCD_4_bit_sendChar(uint8_t u8_a_char);
```



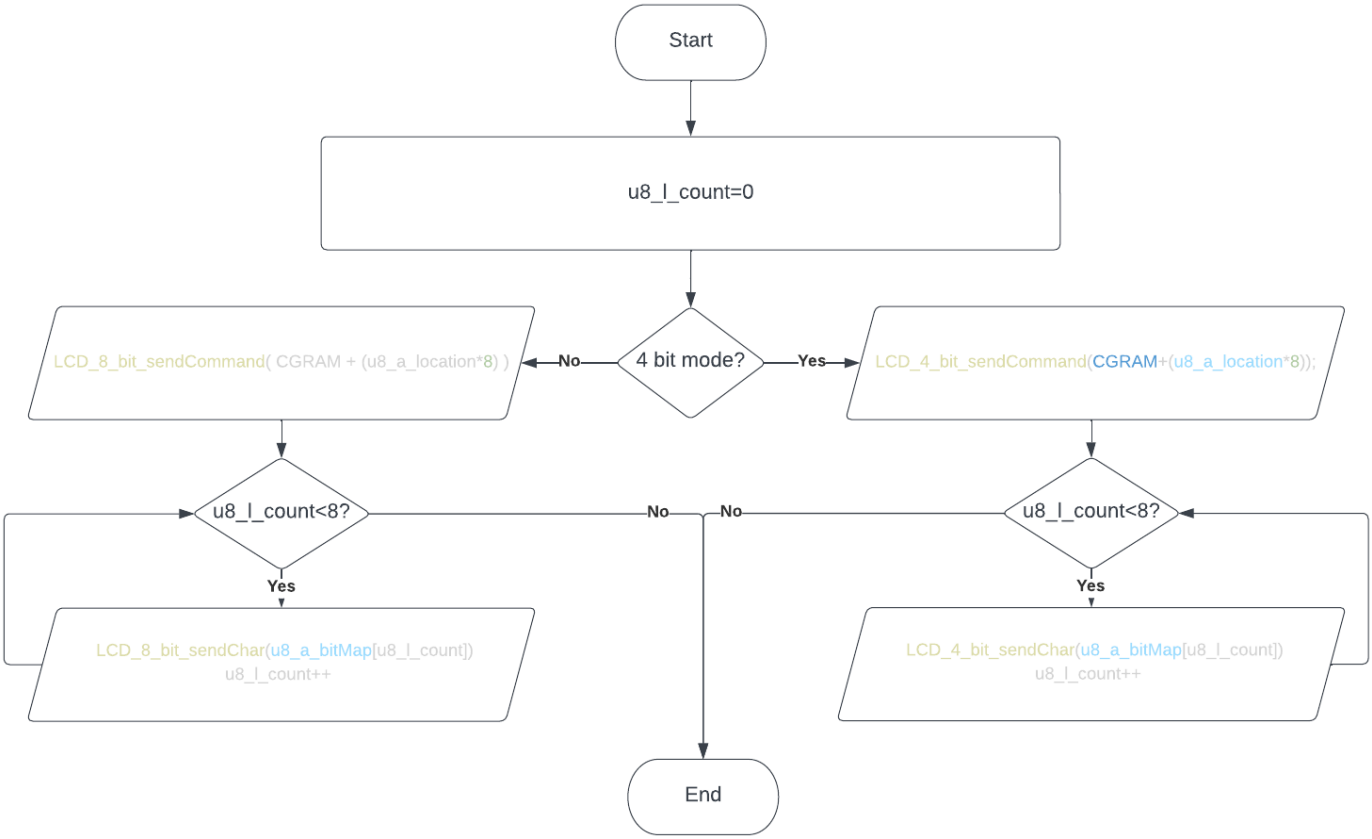
```
LCD_sendString_error LCD_sendString(uint8_t *u8_a_string);
```



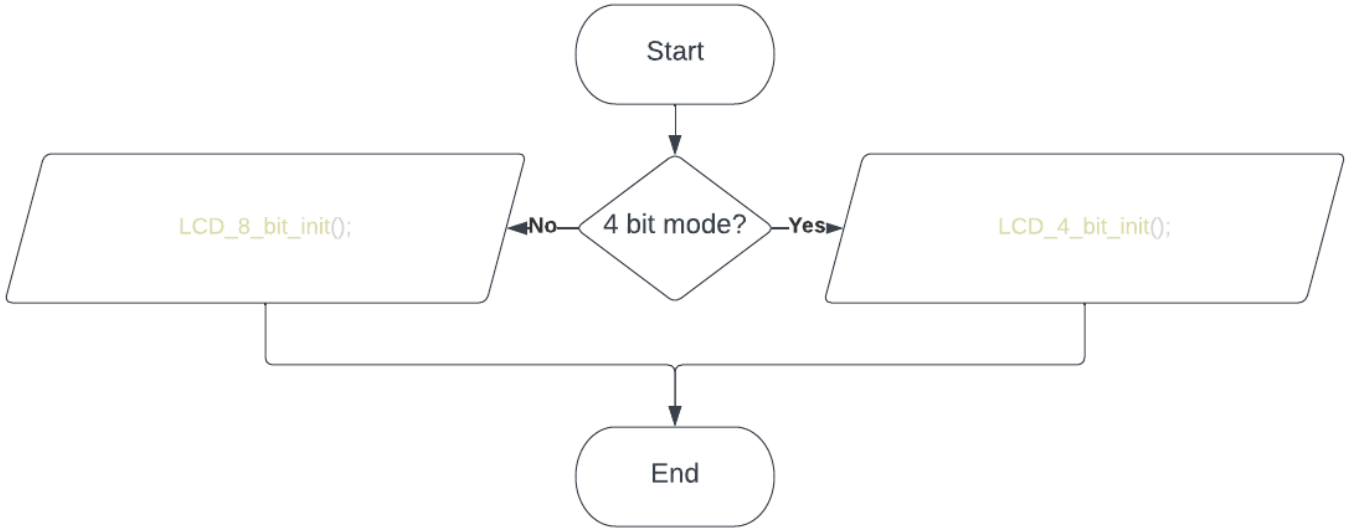
```
void LCD_goTo(uint8_t u8_a_row,uint8_t u8_a_column);
```



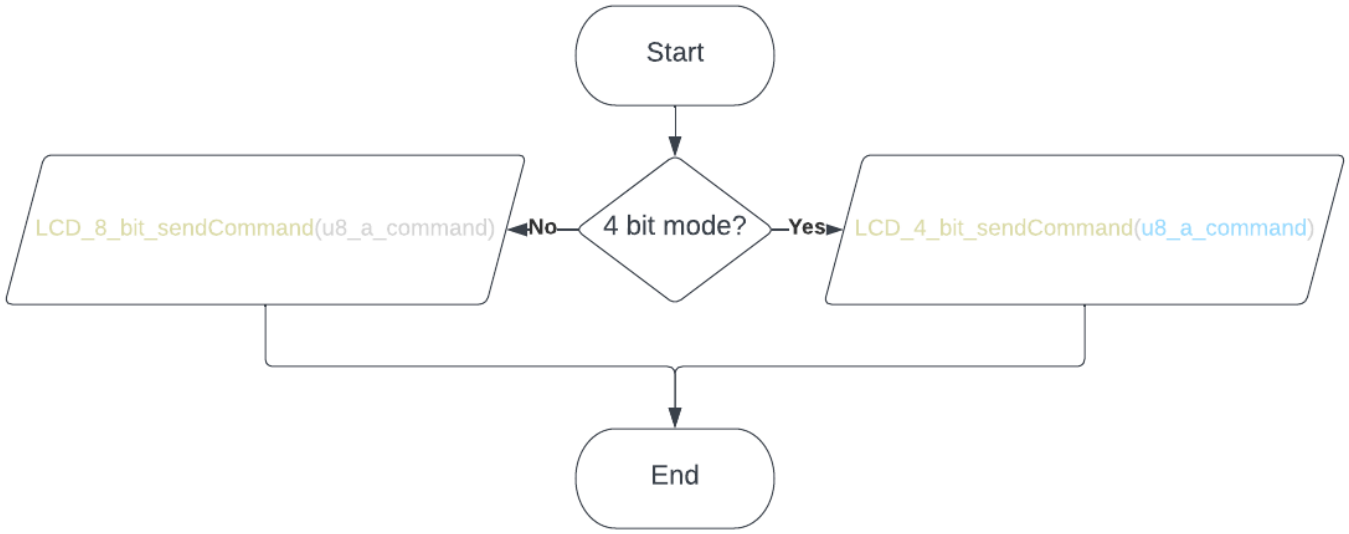
```
void LCD_createCustomCharacter(uint8_t *u8_a_bitMap,uint8_t u8_a_location);
```



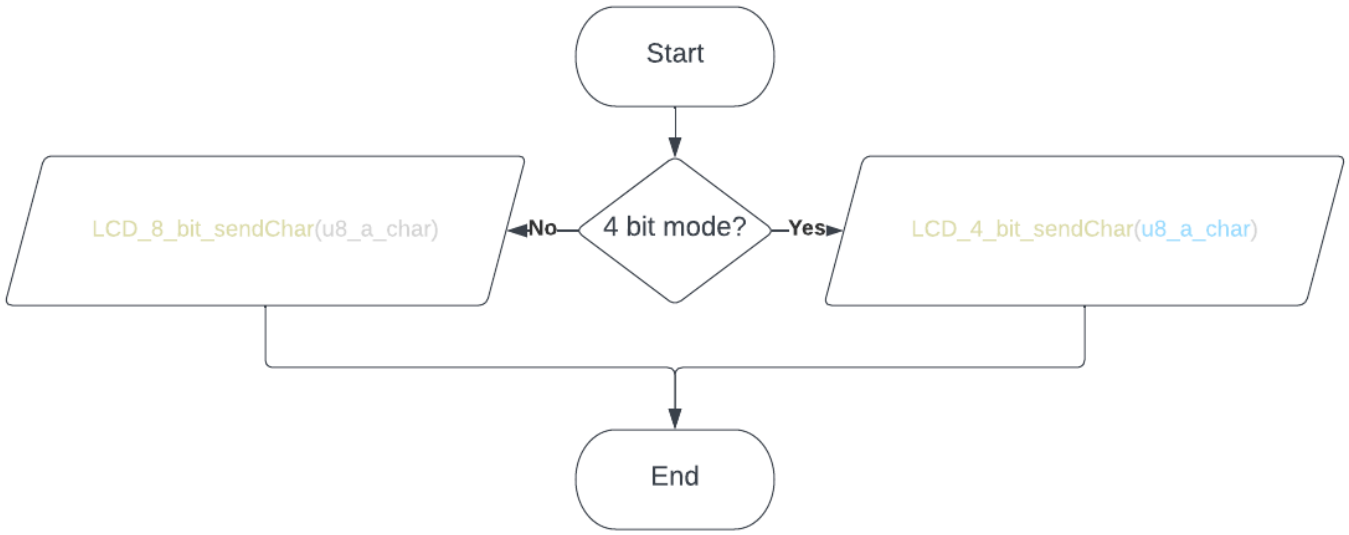
```
LCD_init_error LCD_init(void);
```



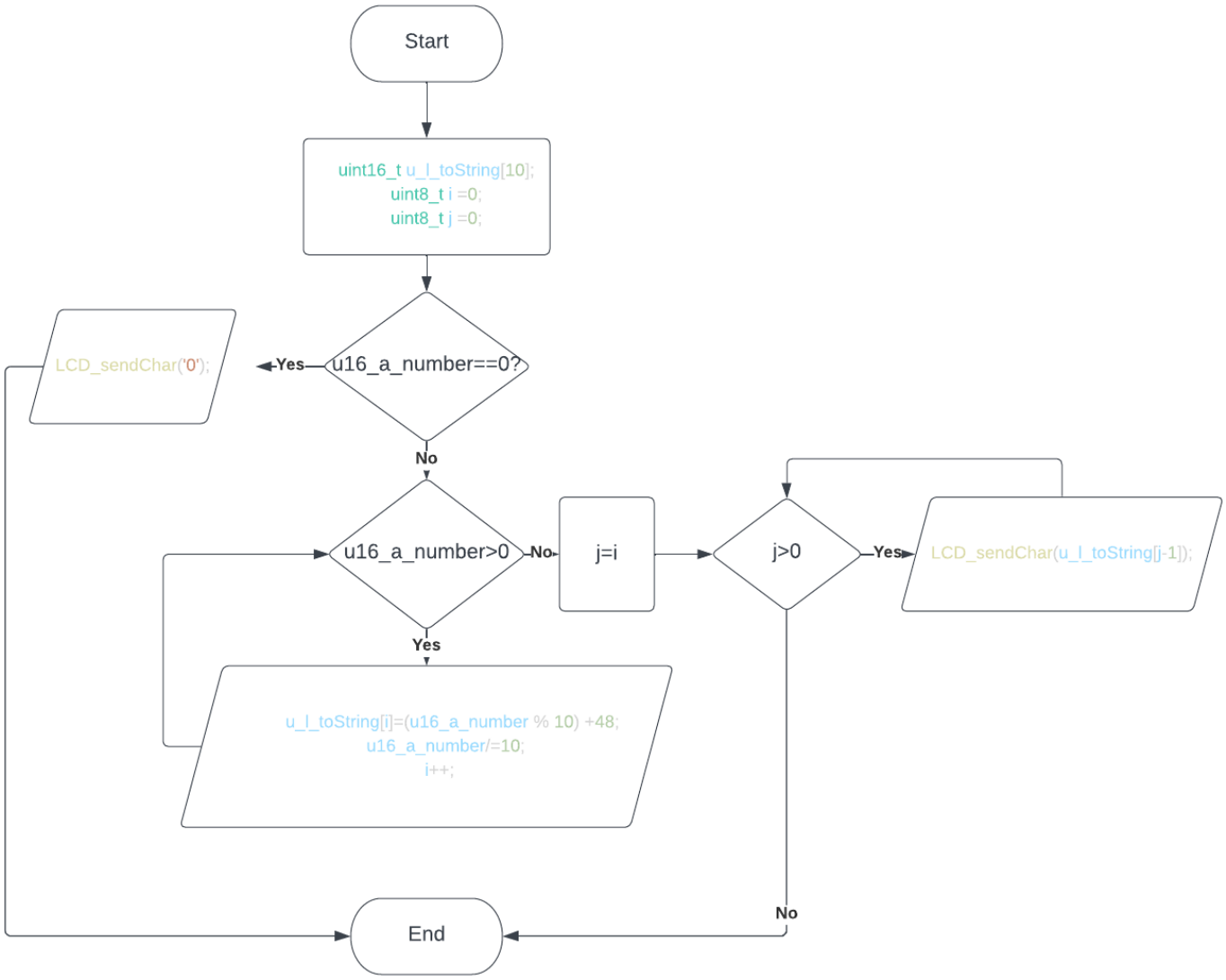

```
LCD_sendCommand_error LCD_sendCommand(uint8_t u8_a_command);
```



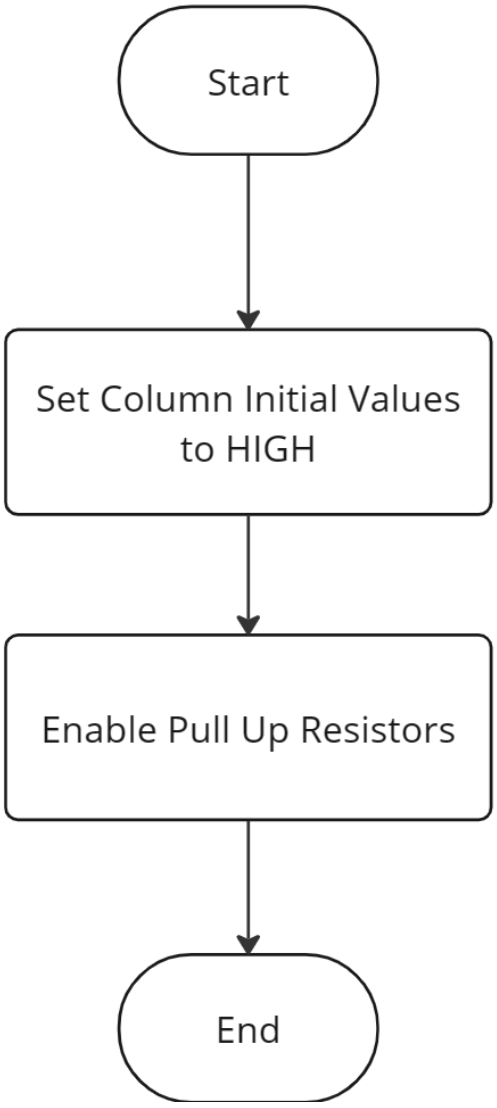
```
LCD_sendChar_error LCD_sendChar(uint8_t u8_a_char);
```



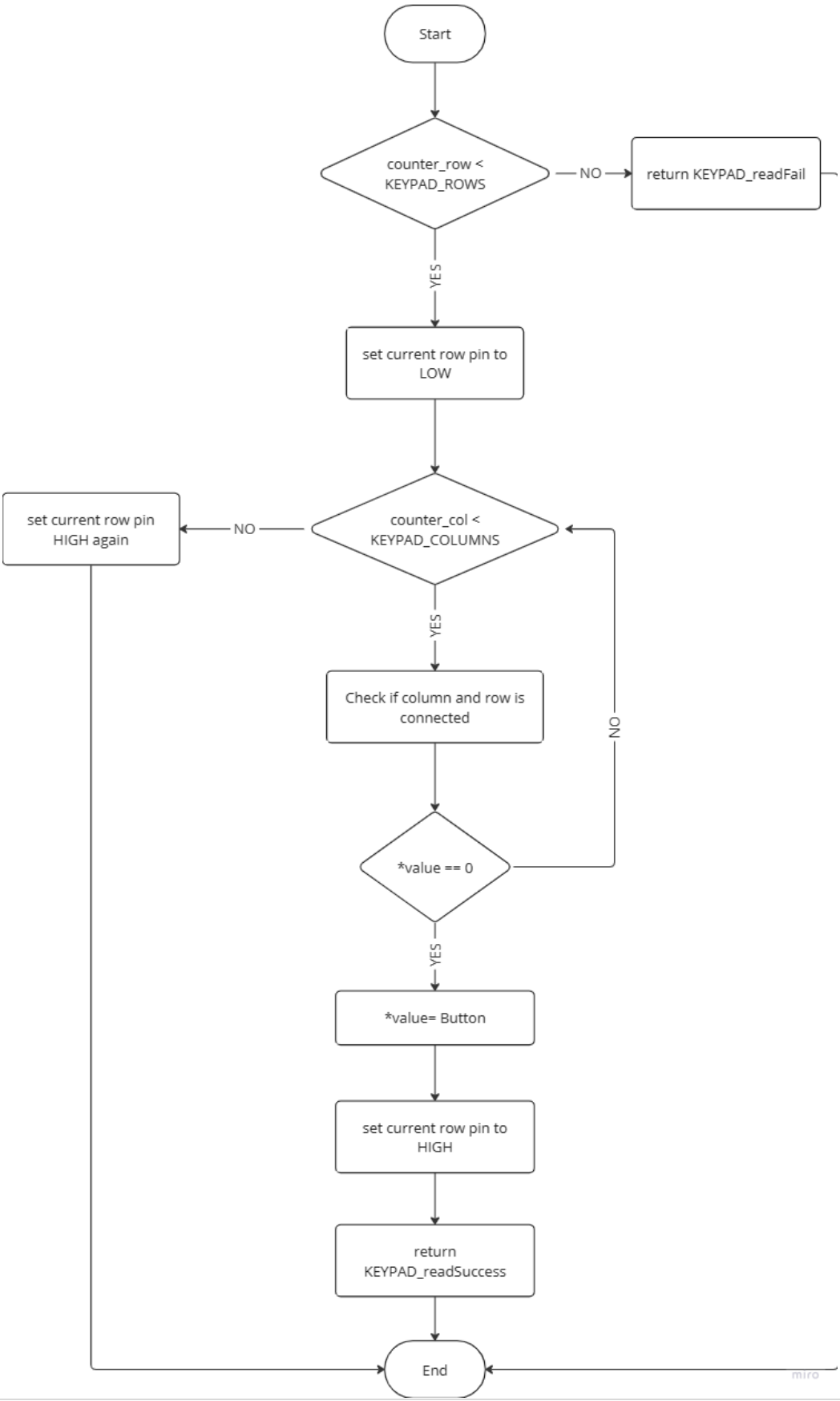
```
sendChar_error LCD_sendInteger(uint16_t u16_a_number);
```



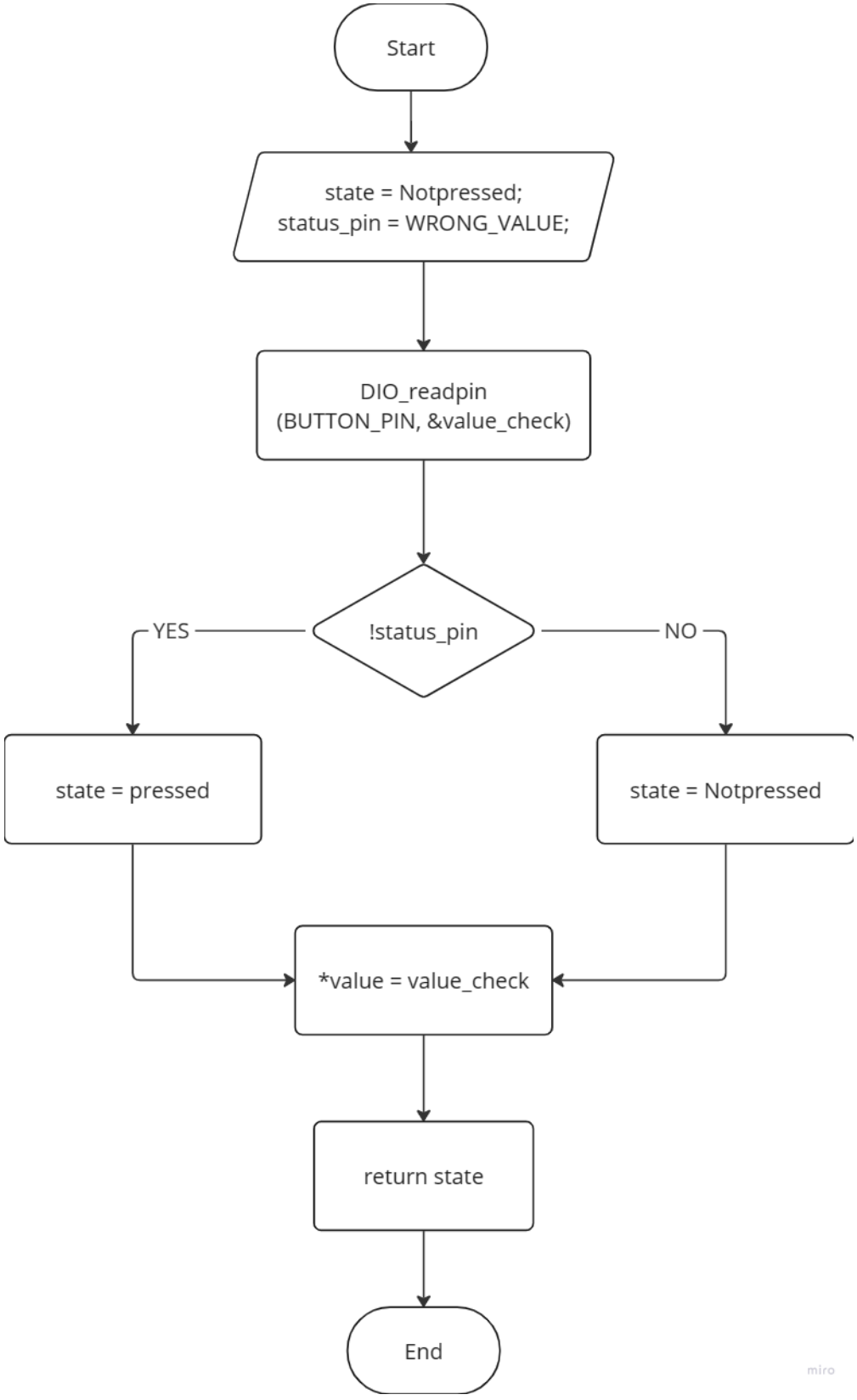
KEYPAD_initError KEYPAD_init(void)



KEYPAD_readError KEYPAD_getpressedkey(u8 *value)



Button


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

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'
```