ATM MACHINE

Team 2

1- Mohab Ahmed

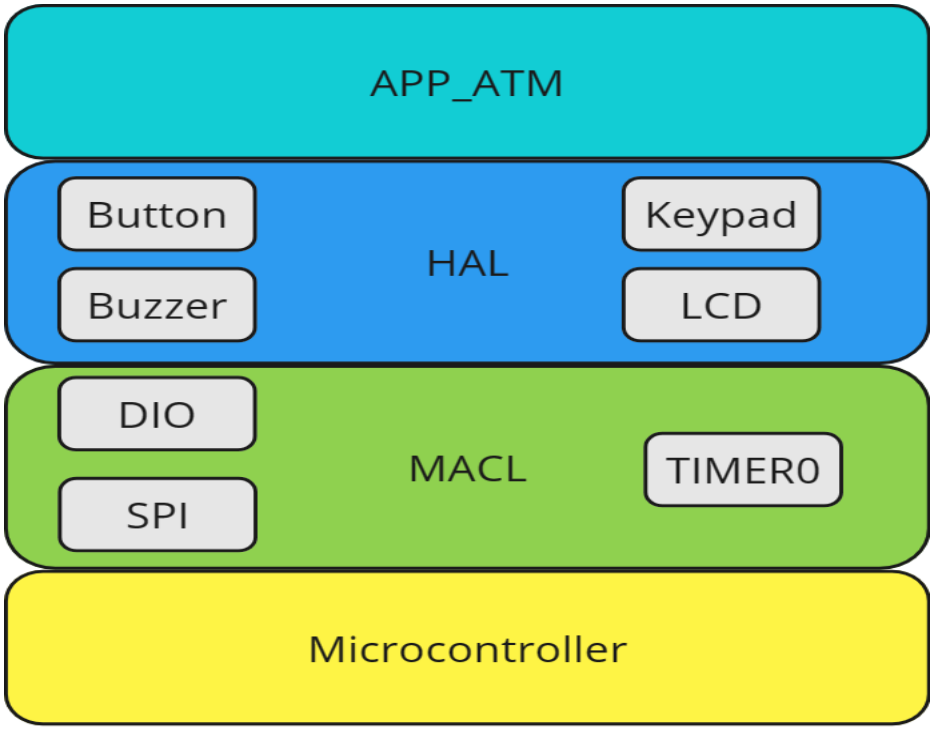
2- Anas Mahmoud

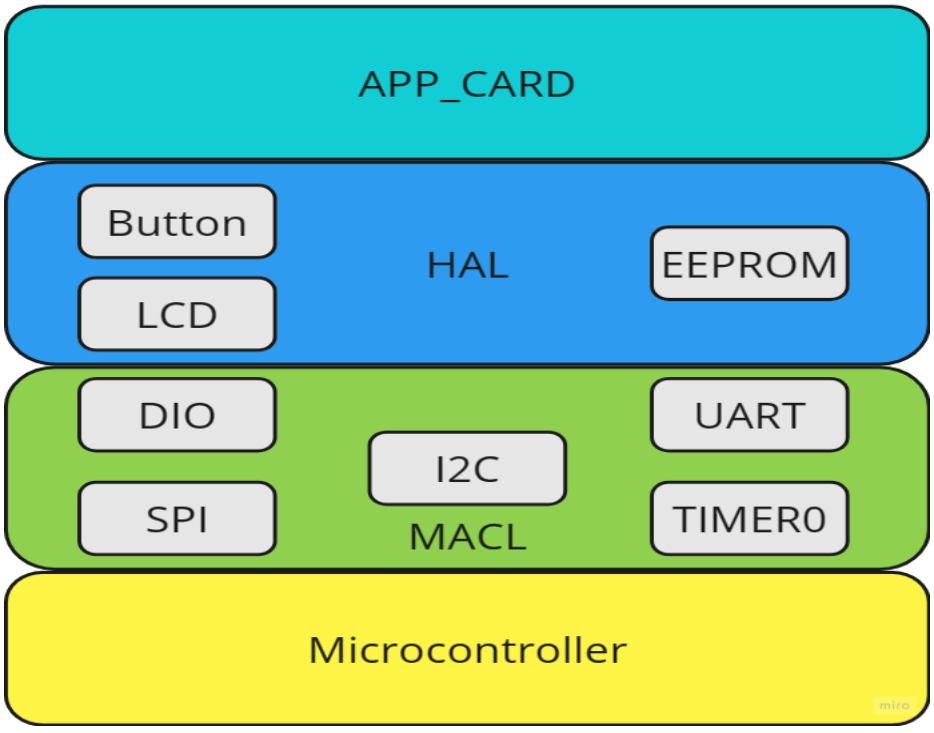
3- Mustafa Mohammed

Table of Contents

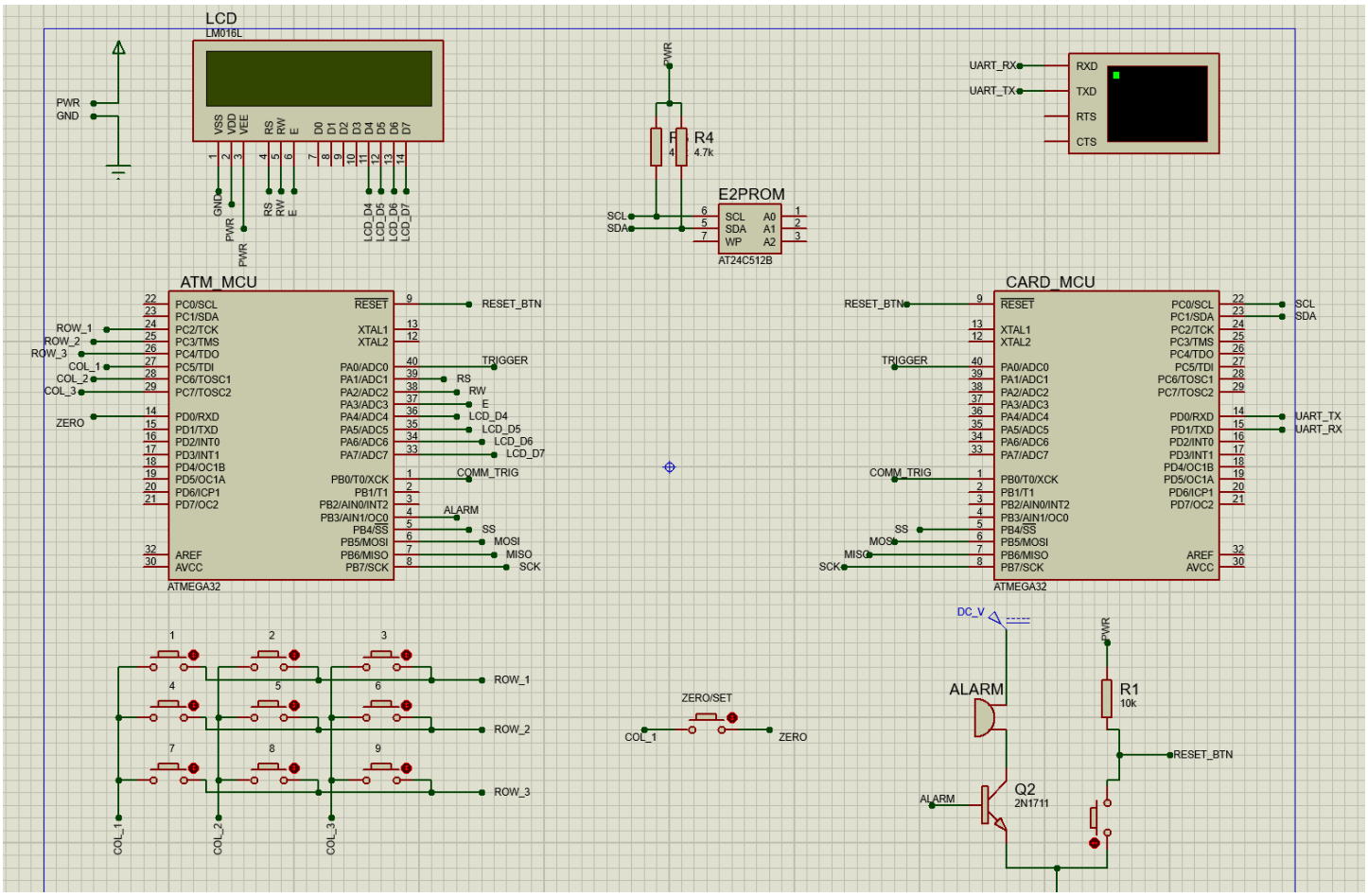
1-High Level Design

1.1 Layered architecture





1.2 Schematic Capture



1.3Drivers Description

1.3.1 DIO Driver

Configuration: Consist of 4 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

1.3.2 Timer Driver

Configuration: Consist of 5 API's

Location: MCAL

Function: used to set a time delay

1.3.3 ADC Driver

Configuration: Consist of 2 API's

Location: MCAL

Function: used to initialize ADC, read the value of ADC

1.3.4 Keypad Driver

Configuration: Consist of 2 API's

Location: HAL

Function: used to initialize the keypad, get pressed key

1.3.5 Button Driver

Configuration: Consist of 2 API's

Location: HAL

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

1.3.6 LCD Driver

Configuration: Consist of 14 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

1.3.7 LM35

Configuration: Consist of 2 API's

Location: HAL

Function: used to initialize LM35 sensor, get the temperature value

1.3.8 Application Driver

Configuration: Consist of 9 API's

Location: App

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

1.4 API's

1.4.1 DIO Driver

1- PinDirection_t DIO_setpindir (uint8_t u8_a_portid, uint8_t u8_a_pinid,

uint8_t u8_a_pindir);

2- PinValue_t DIO_setpinvalue (uint8_t u8_a_portid, uint8_t u8_a_pinid, uint8_t u8_a_pinval);

3- PinRead_t DIO_readpin (uint8_t u8_a_portid, uint8_t u8_a_pinid, uint8_t* u8_a_val);

4- PinRead_t DIO_togglepin (uint8_t u8_a_portid, uint8_t u8_a_pinid);

1.4.2 Timer0 Driver

1- TMR0_init_error TMR0_init(void);

2- TMR0_start_error TMR0_start(void);

3- TMR0_stop_error TMR0_stop(void);

4- TMR0_delay_error TMR0_delayms(uint32_t u32_a_delayms);

5- TMR0_delay_error TMR0_delaymicos(uint32_t u32_a_delaymicros);

1.4.3 ADC Driver

1- ADC_initstatus ADC_Init(void);

2- uint16_t ADC_read(void);

1.4.4 Keypad Driver

1- void KEYPAD_init(void) ;

2- uint8_t KEYPAD_getpressedkey(void) ;

1.4.5 Button Driver

1- void Button_init(uint8_t u8_a_Buttonport , uint8_t u8_a_ Button_pin);

2- button_t Ispressed(uint8_t u8_a_Buttonport , uint8_t u8_a_Button_pin ,

uint8_t * u8_pvalue);

1.4.6 LCD Driver

1- LCD_init_error LCD_8_bit_init (void);

2- LCD_sendCommand_error LCD_8_bit_sendCommand(uint8_t u8_a_command);

3- LCD_sendChar_error LCD_8_bit_sendChar(uint8_t u8_a_char);

4- LCD_init_error LCD_4_bit_init(void);

5- LCD_sendCommand_error LCD_4_bit_sendCommand(uint8_t u8_a_command);

6- LCD_sendChar_error LCD_4_bit_sendChar(uint8_t u8_a_char);

7- LCD_sendString_error LCD_sendString(uint8_t *u8_a_string);

8- void LCD_goTo(uint8_t u8_a_row,uint8_t u8_a_column);

9- void LCD_createCustomCharacter(uint8_t *u8_a_bitMap,uint8_t u8_a_location)

10- LCD_init_error LCD_init(void);

11- LCD_sendCommand_error LCD_sendCommand(uint8_t u8_a_command);

12- LCD_sendChar_error LCD_sendChar(uint8_t u8_a_char);

13- LCD_sendChar_error LCD_sendFloat(float f_a_number);

14- LCD_sendChar_error LCD_sendInteger(uint16_t u16_a_number);

1.4.7 LM35 Driver

1- LM35_status LM35_init(void);

2-LM35_status LM35_init(void);

1.4.8 SPI Driver

1-en_a_spierrstatus SPI_initmaster(void);

2-en_a_spierrstatus SPI_initslave(void);

3-en_a_spierrstatus SPI_masterinittransmit(void);

4-en_a_spierrstatus SPI_masterendtransmit(void);

5-en_a_spierrstatus SPI_sendbyte(uint8_t u8_a_data);

6-en_a_spierrstatus SPI_receivebyte(uint8_t * u8_a_recdata);

7-en_a_spierrstatus SPI_sendstring(const uint8_t * u8_a_str);

8-en_a_spierrstatus SPI_receivestring(uint8_t * u8_a_str);

1.4.9 I2C Driver

1- void TWI_init(void);

2-void TWI_setaddress(uint8_t u8_a_address);

3-void TWI_start(void);

4-void TWI_repeatedstart(void);

5-uint8_t TWI_readwithack(void);

6-uint8_t TWI_readwithnack(void);

7-void TWI_write(uint8_t u8_a_data);

8-void TWI_stop(void);

9-uint8_t TWI_getstatus(void);

1.4.10 UART Driver

1-uart_errorstatus UART_init(void);

2-uart_errorstatus UART_sendchar(uint8_t ua_a_data);

3-uart_errorstatus UART_sendstr(uint8_t * ua_a_string);

4-uart_errorstatus UART_receivechar(uint8_t * u8_a_recdata);

5-uart_errorstatus UART_receivestr(uint8_t * u8_a_recstring);

1.4.11 APP_CARD Driver

1-void APP_init(void);

2-uint8_t APP_entrypoint(void);

3-void APP_sendcarddata(void) ;

4-uint8_t APP_cardprogram(void);

5-void APP_cardfailed(void);

6-void APP_storecard(void);

7-void APP_getcarddata(void);

8-APP_sendtrigger();

1.4.12 APP_ATM Driver

1-void APP_init(void);

2-void APP_readuserpin(void);

3-void APP_startcardcomm(void);

4-void APP_getamount(void);

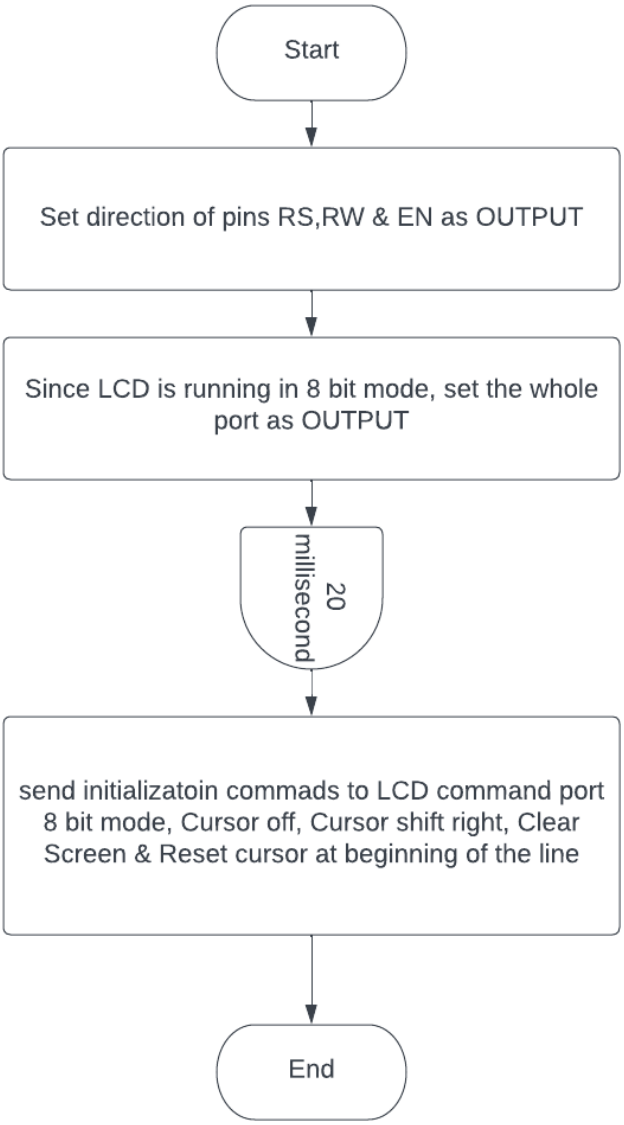
5-void APP_cardvalidate(void);

2- Low Level Design

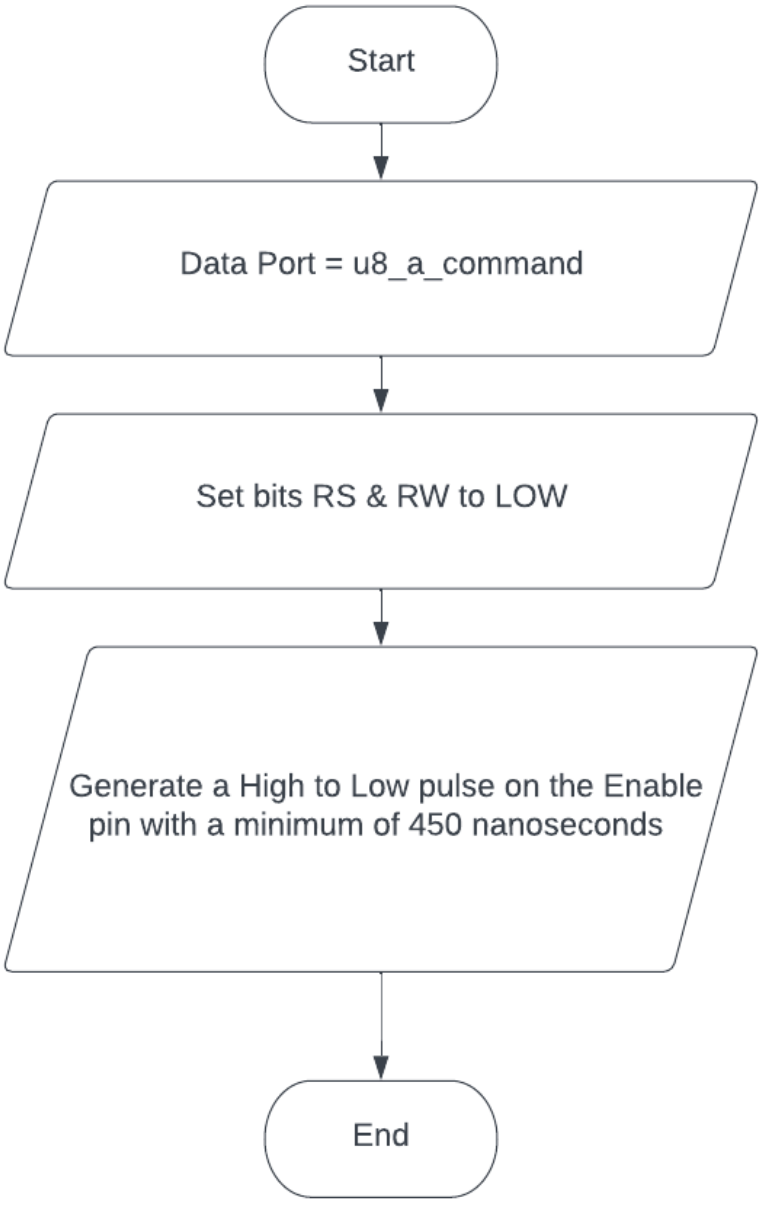
2.1- API's Flow Chart

2.1.1 LCD

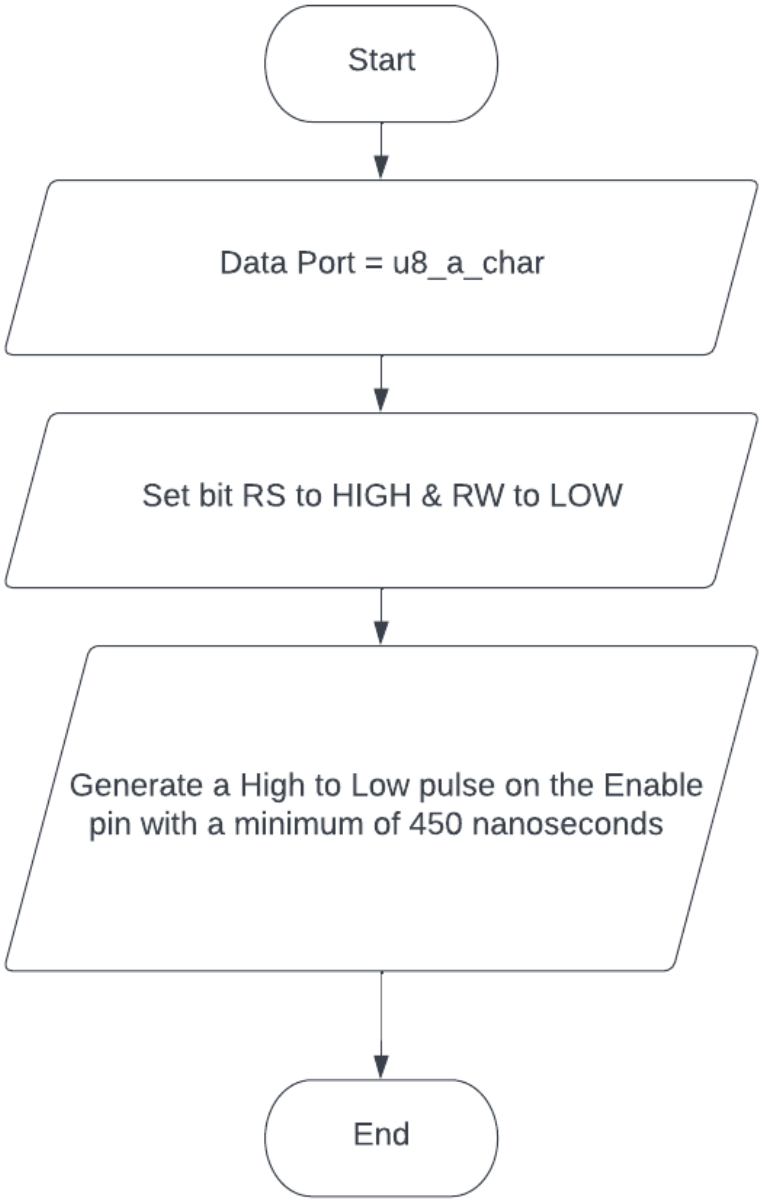
```
LCD_init_error LCD_8_bit_init(void);
```



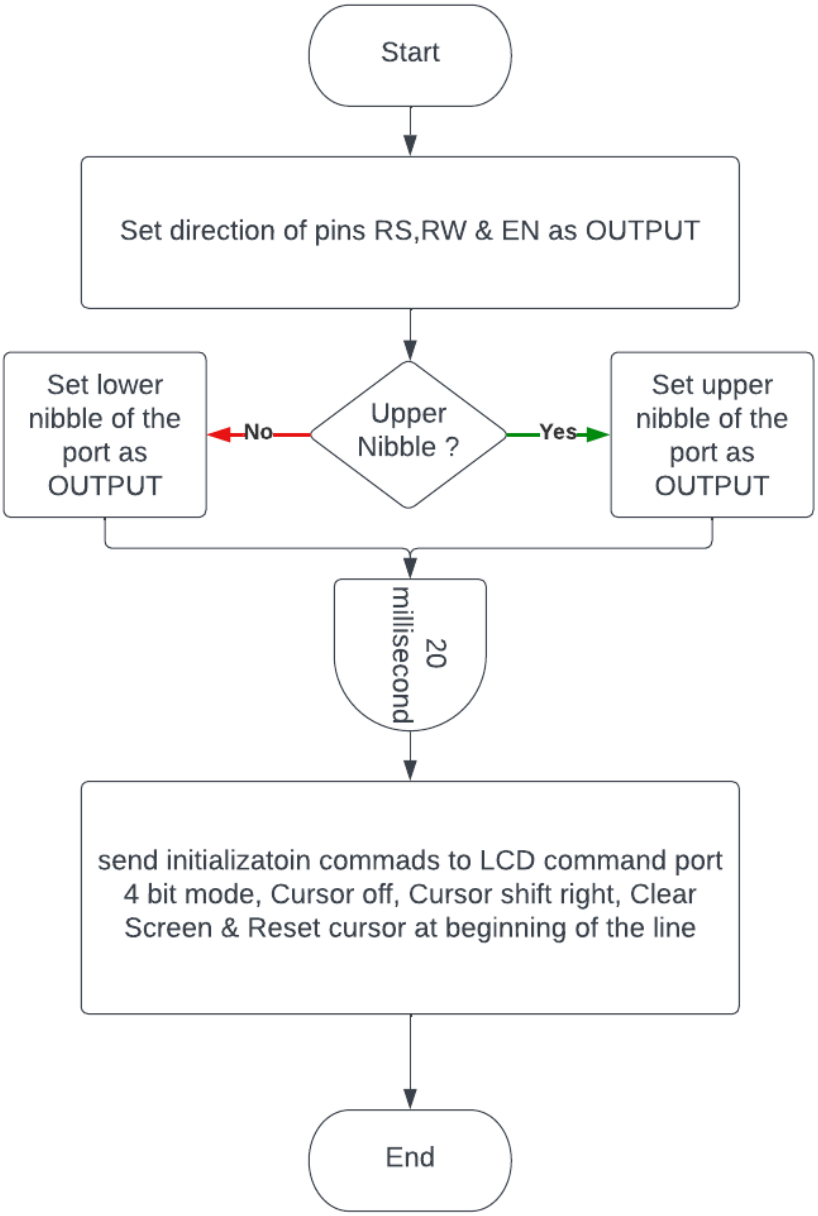
```
LCD_sendCommand_error LCD_8_bit_sendCommand(uint8_t u8_a_command);
```



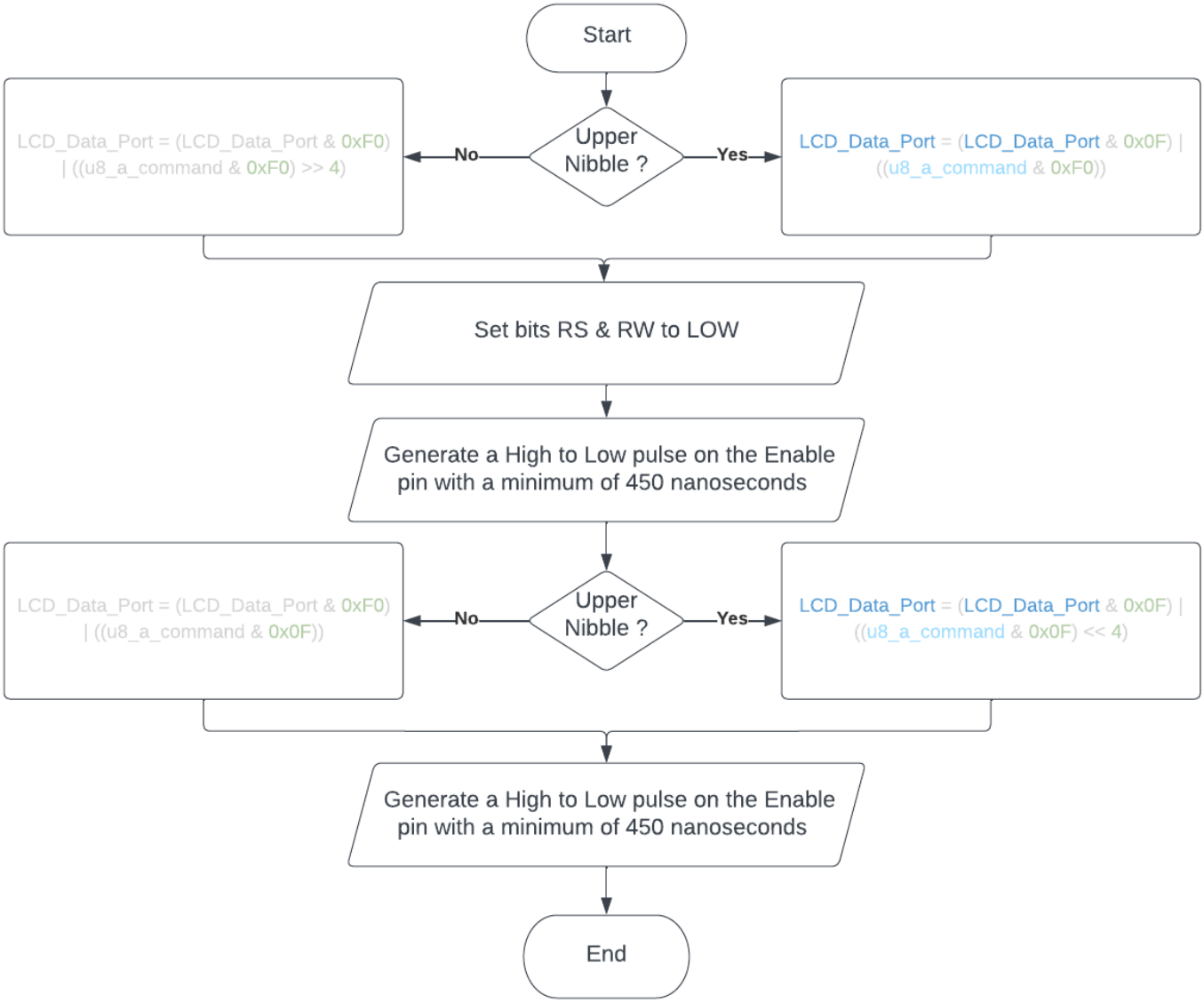
```
LCD_sendChar_error LCD_8_bit_sendChar(uint8_t u8_a_char);
```



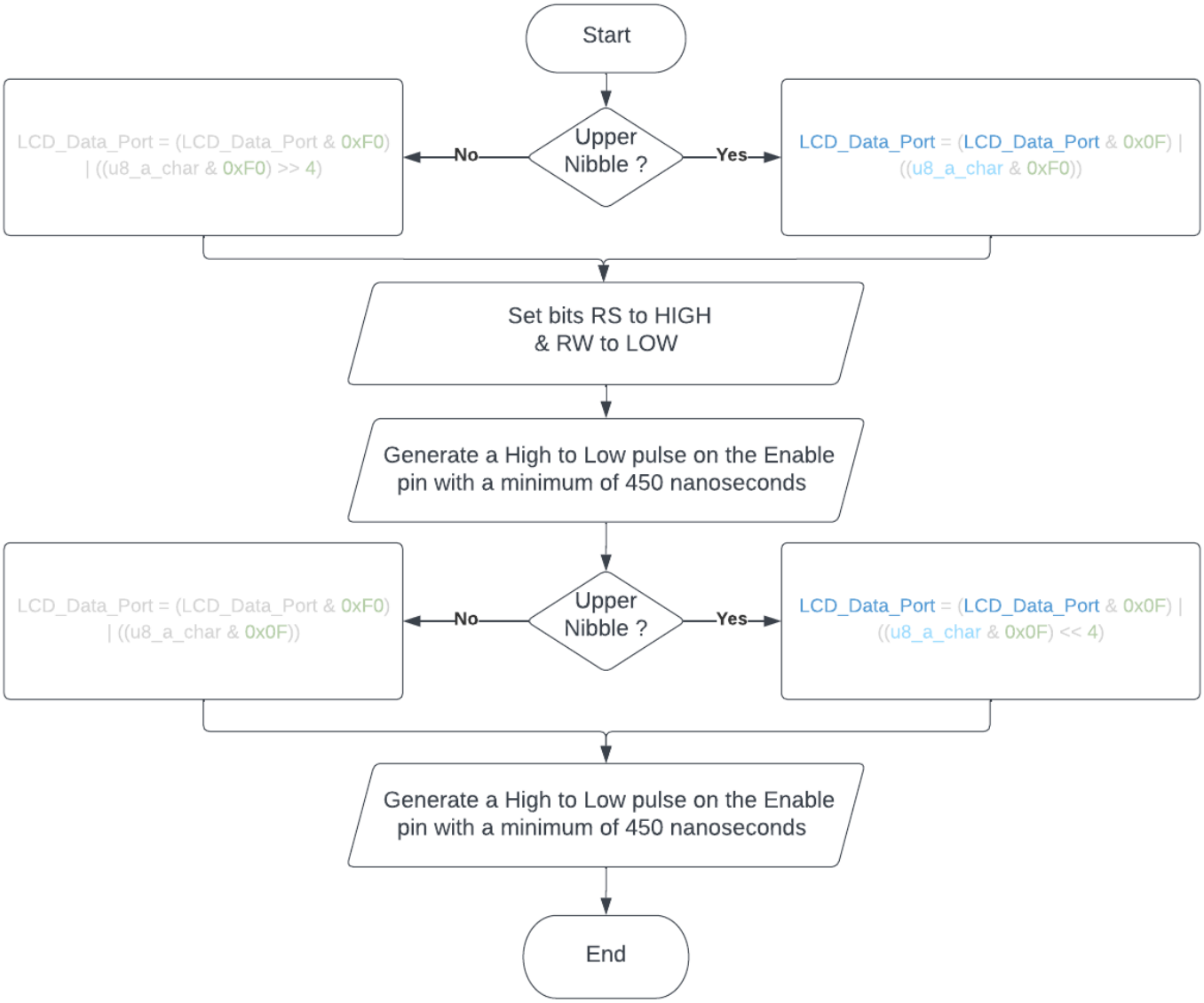
```
LCD_init_error LCD_4_bit_init(void);
```



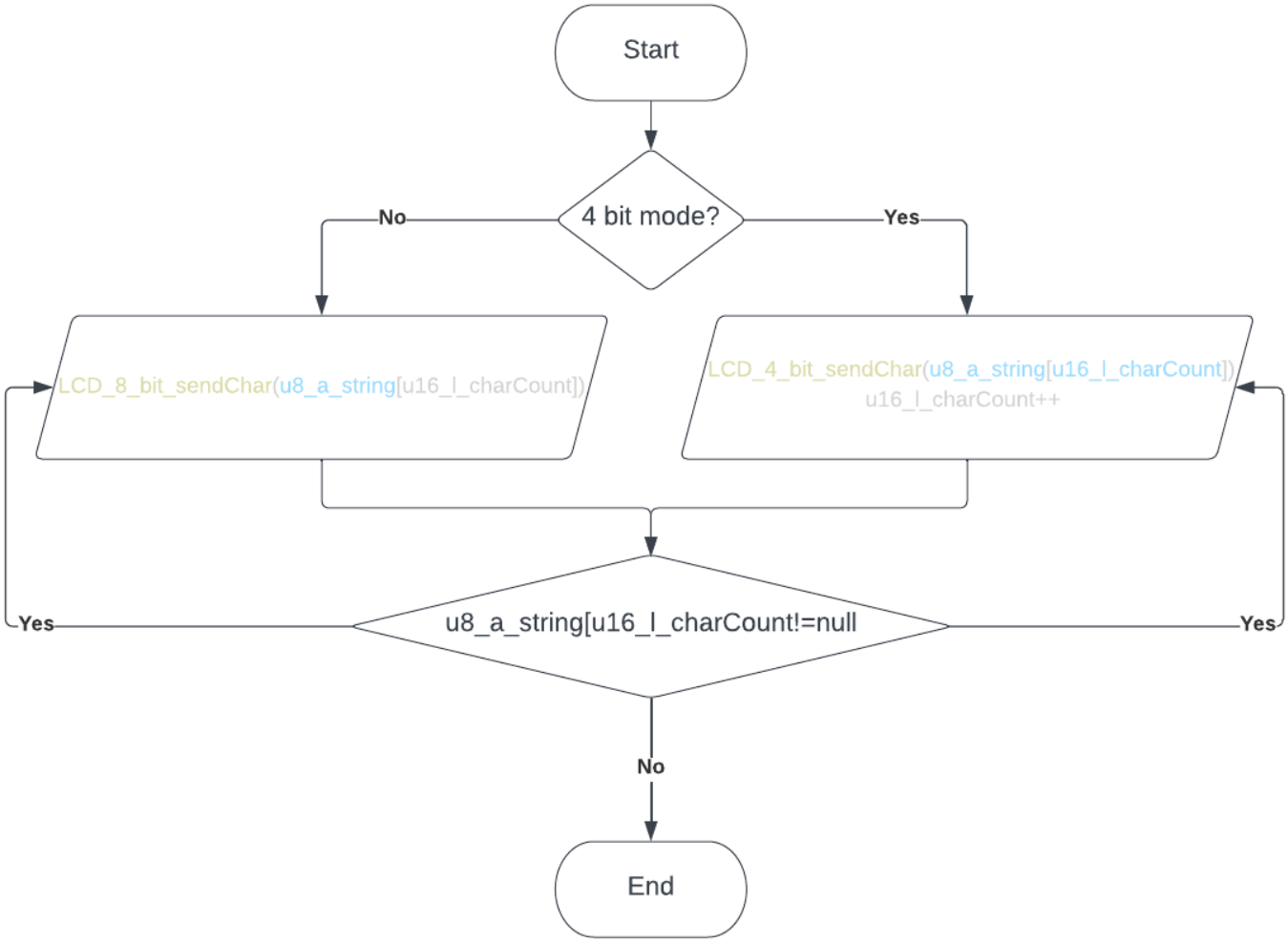
```
LCD_sendCommand_error LCD_4_bit_sendCommand(uint8_t u8_a_command);
```



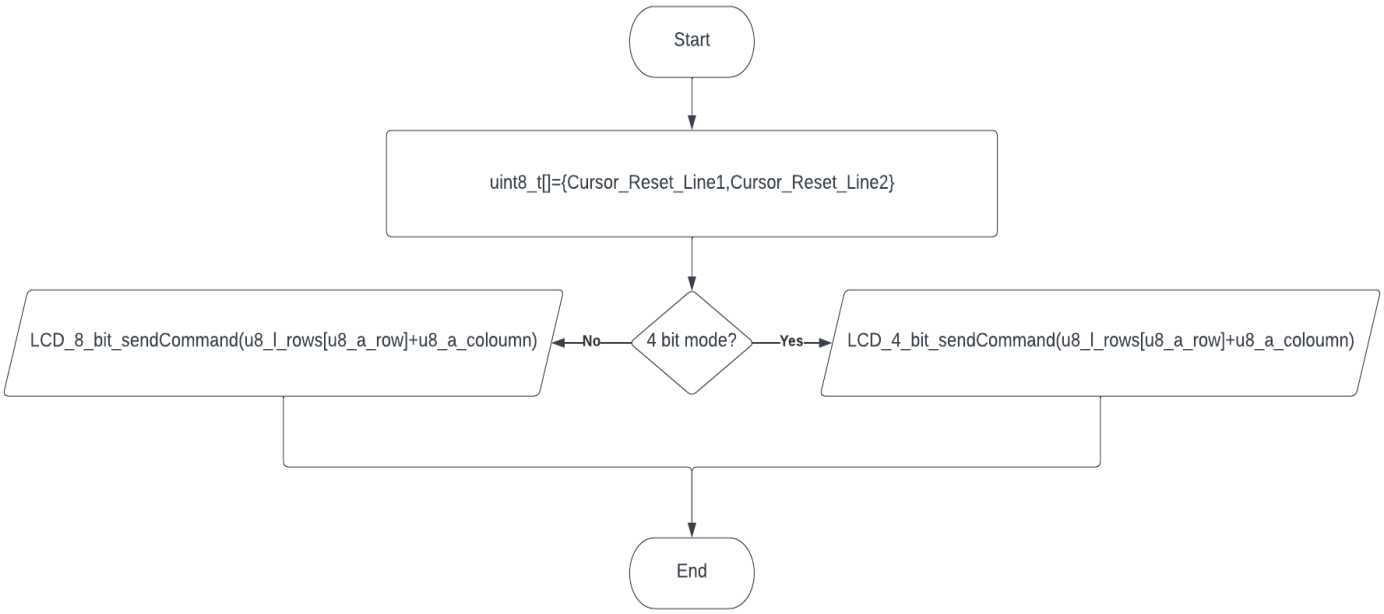
```
LCD_sendChar_error LCD_4_bit_sendChar(uint8_t u8_a_char);
```



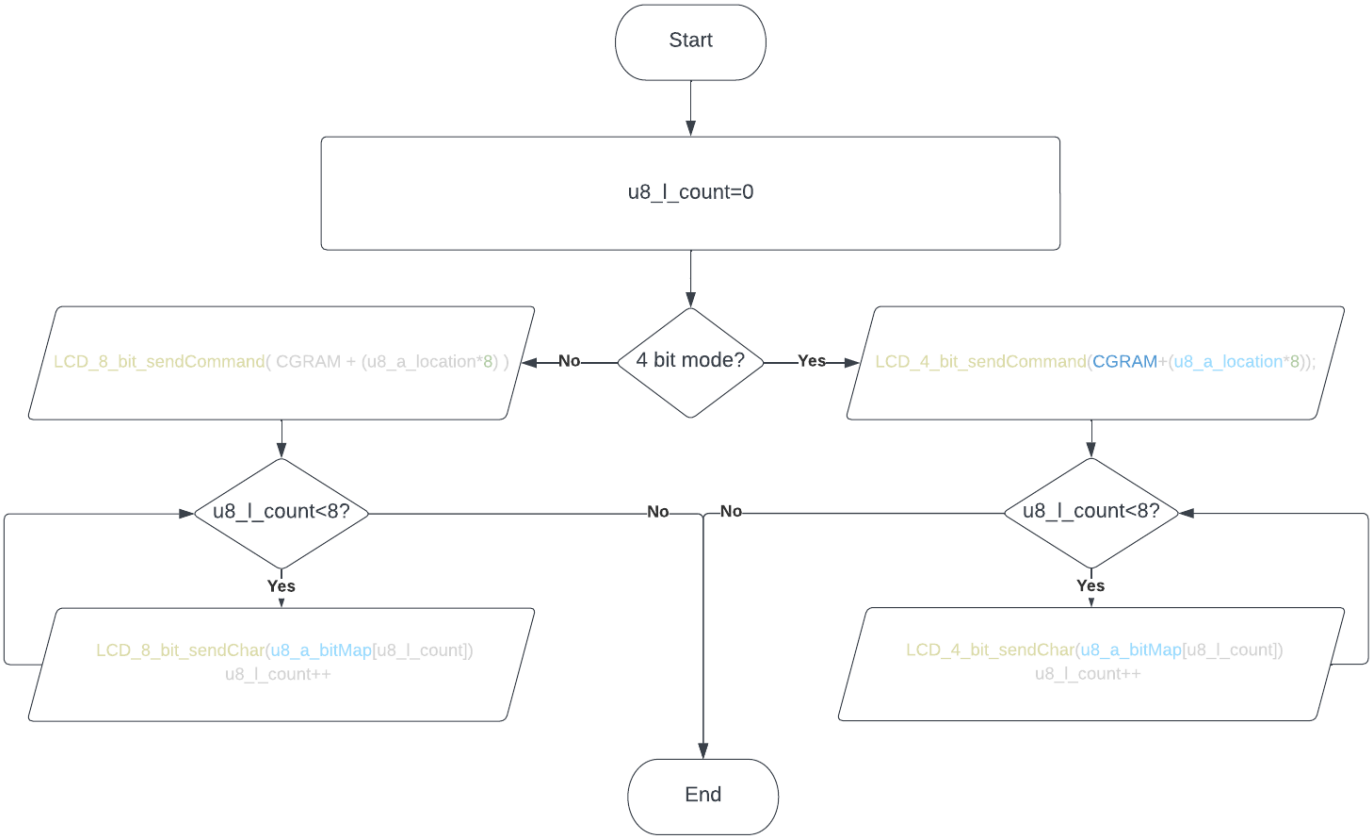
```
LCD_sendString_error LCD_sendString(uint8_t *u8_a_string);
```



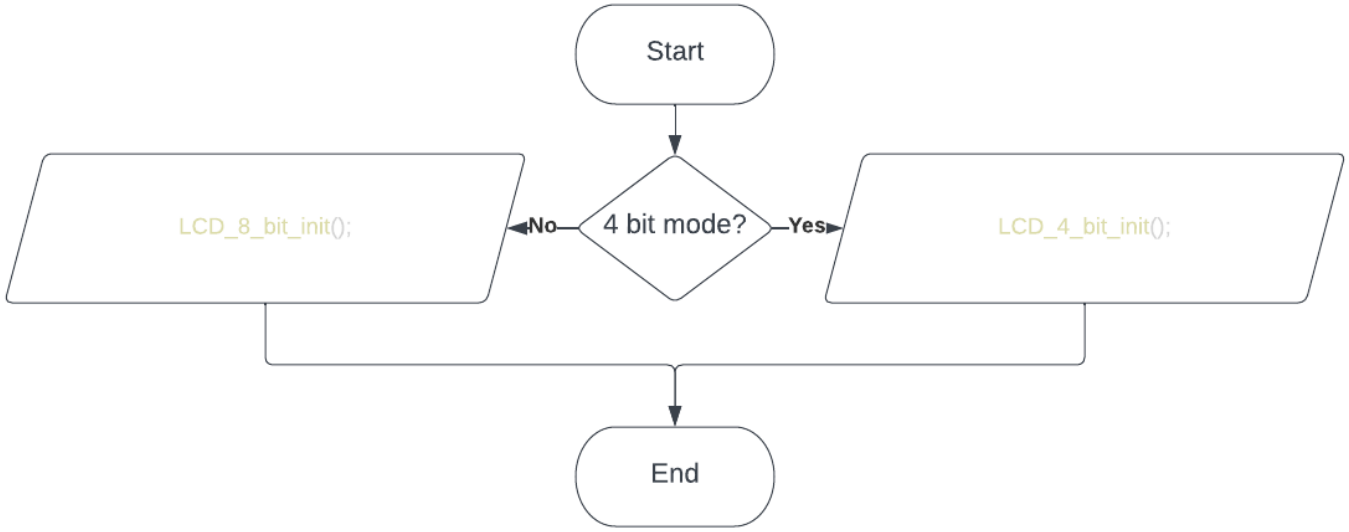
```
void LCD_goTo(uint8_t u8_a_row,uint8_t u8_a_column);
```



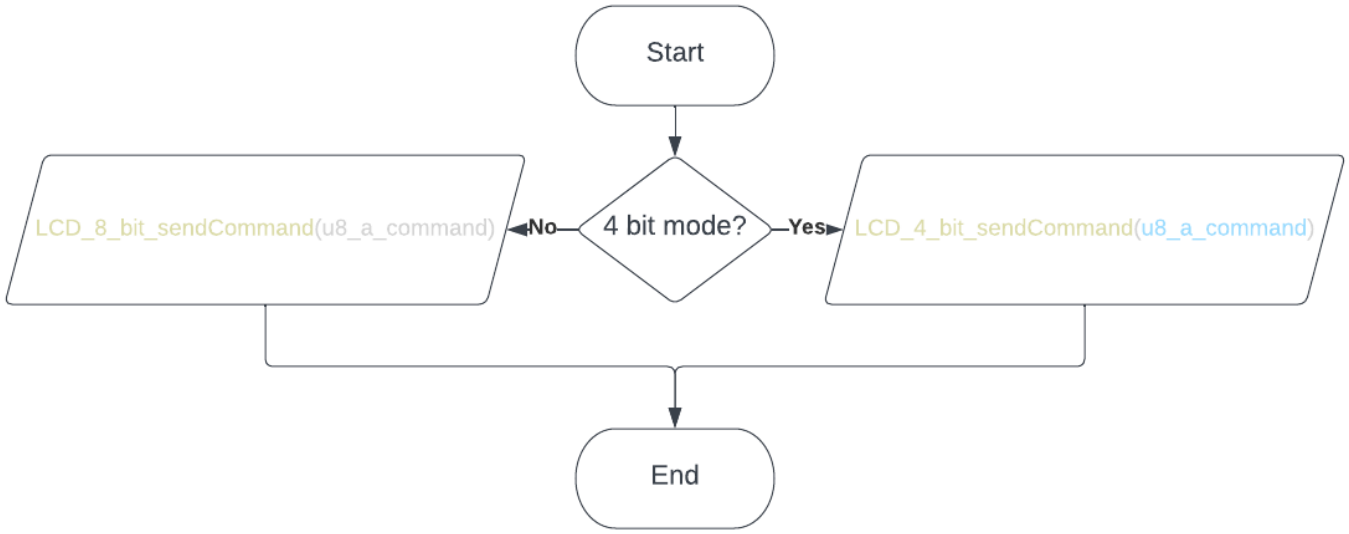
```
void LCD_createCustomCharacter(uint8_t *u8_a_bitMap,uint8_t u8_a_location);
```



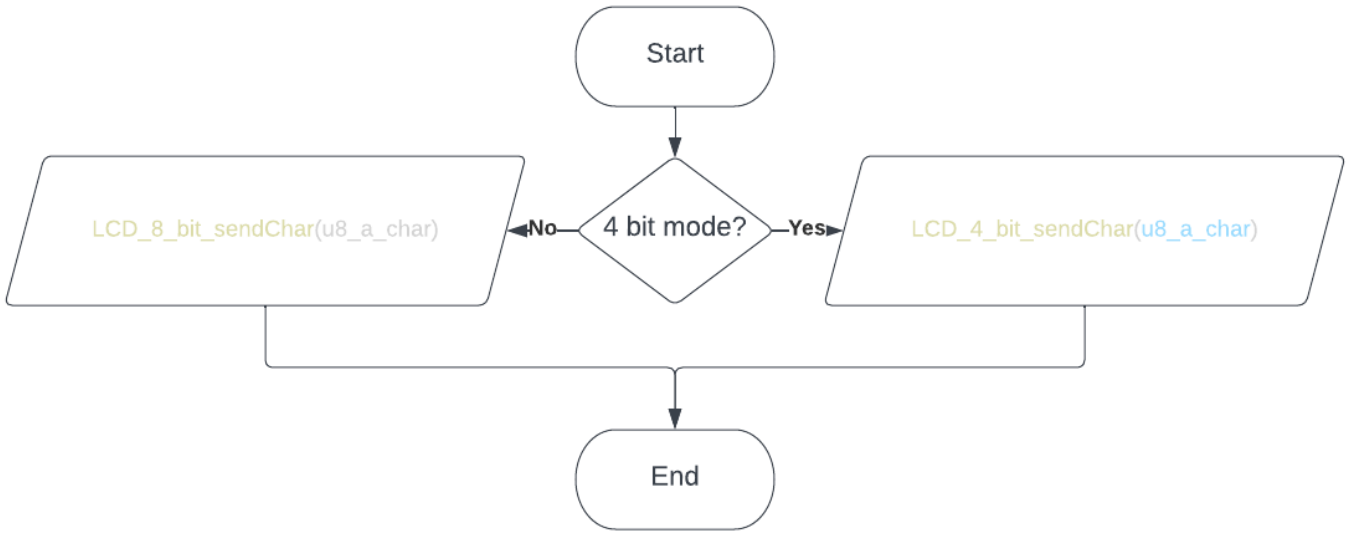
```
LCD_init_error LCD_init(void);
```



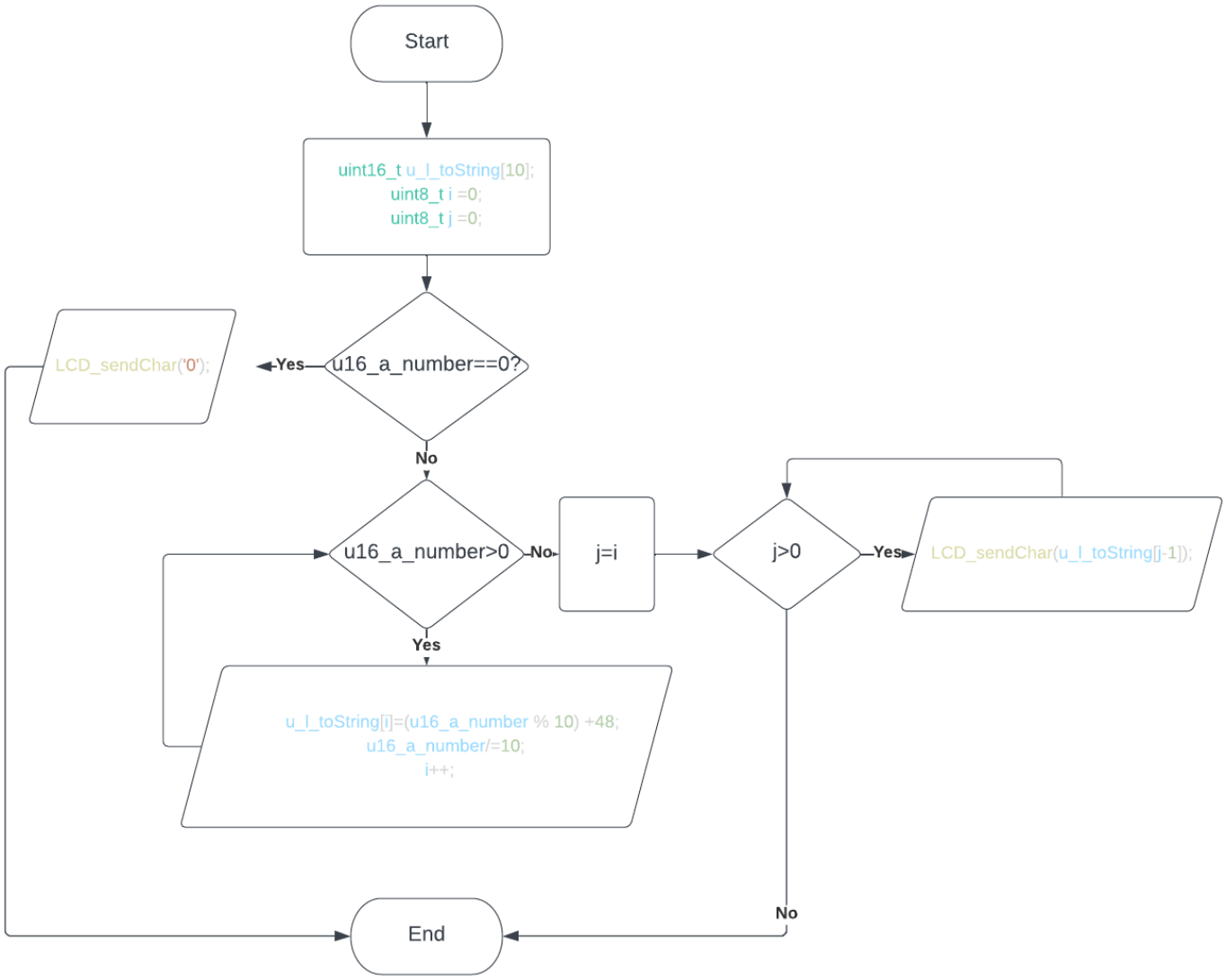

```
LCD_sendCommand_error LCD_sendCommand(uint8_t u8_a_command);
```



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

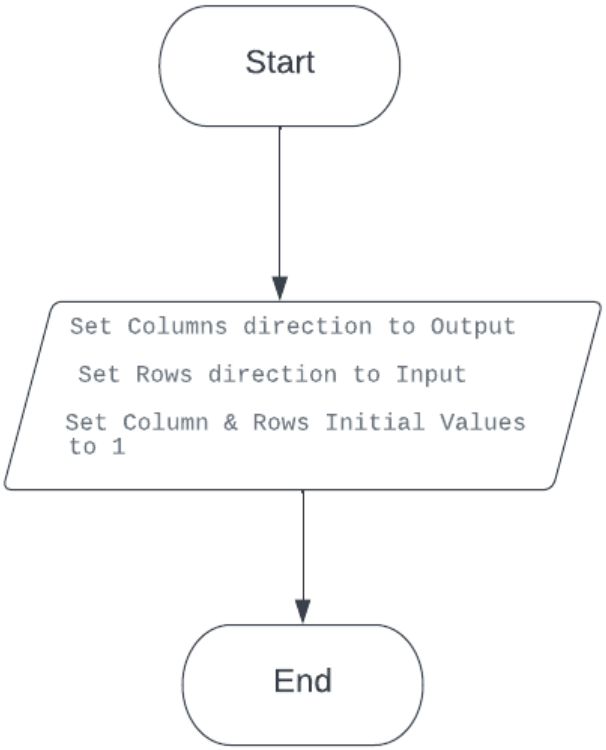


```
sendChar_error LCD_sendInteger(uint16_t u16_a_number);
```

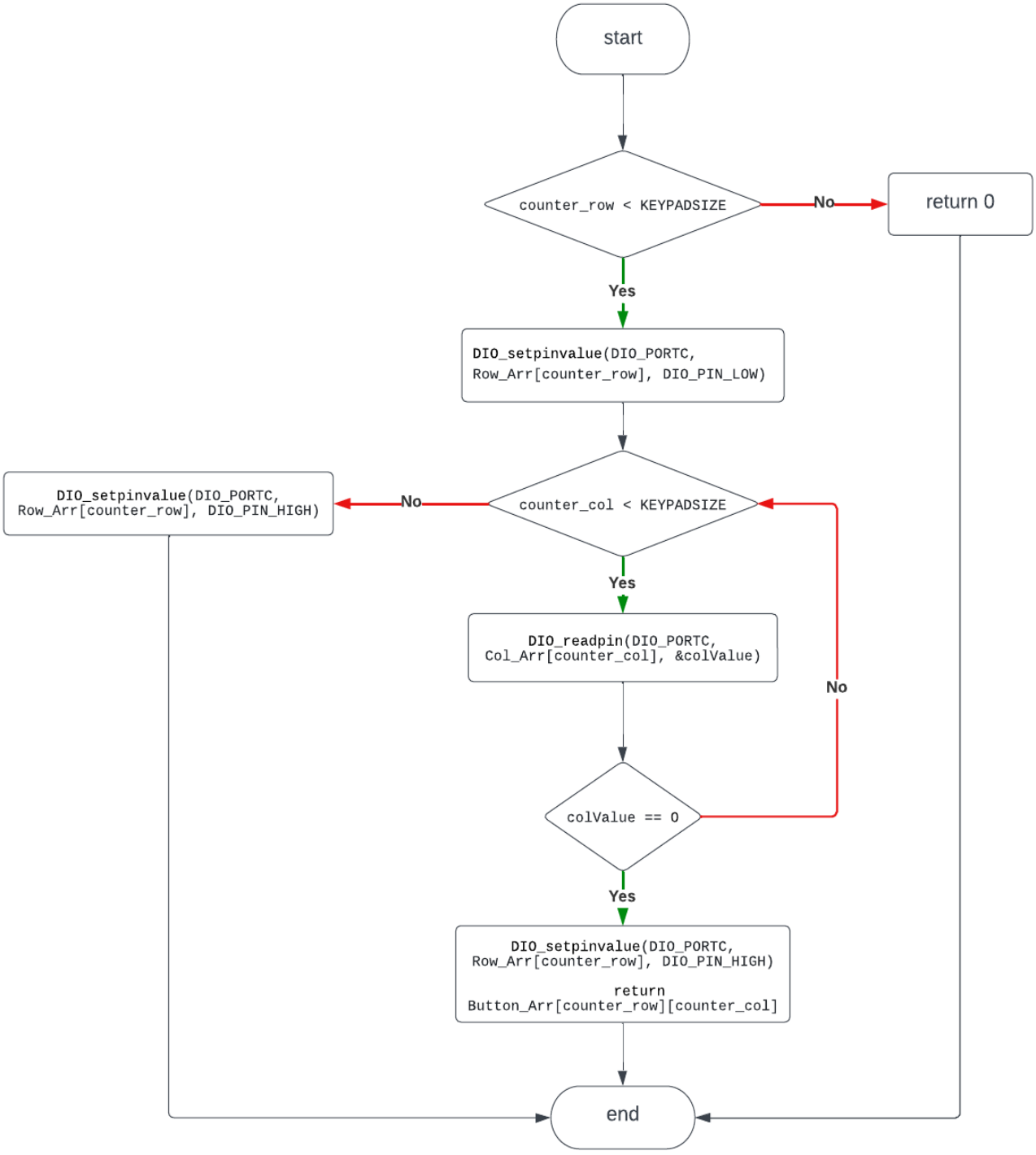


2.1.2 Keypad

```
void KEYPAD_init(void)
```

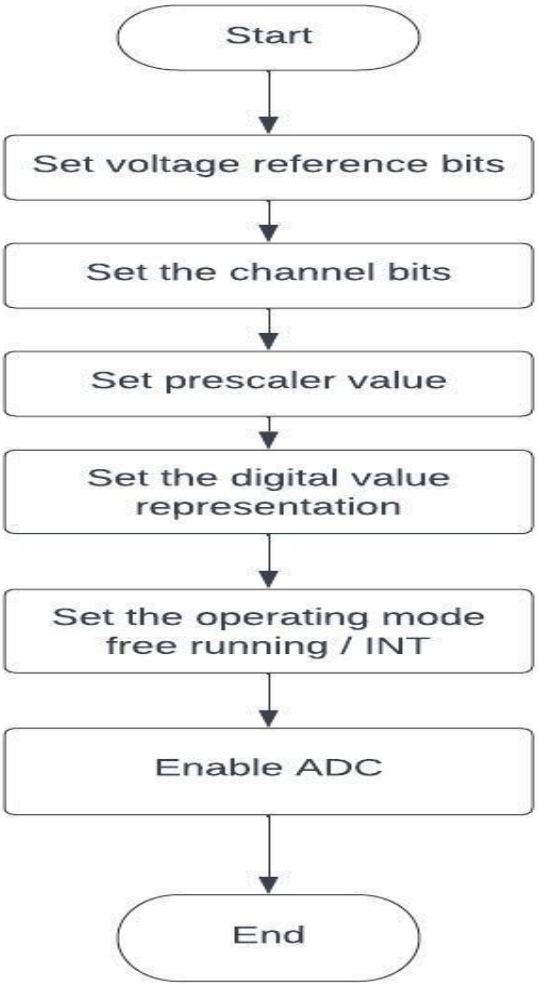


```
uint8_t KEYPAD_getpressedkey(void)
```

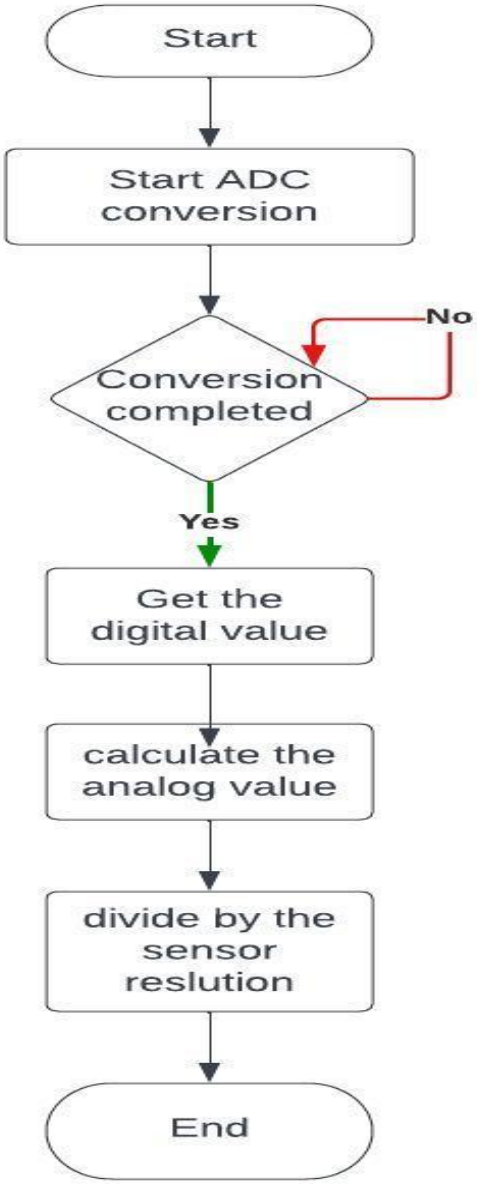


2.1.3 ADC

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

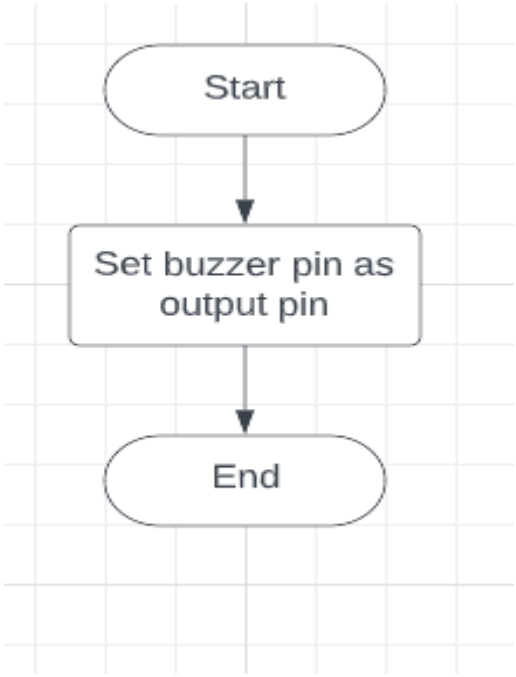


```c
uint32_t ADC_read(void);
```

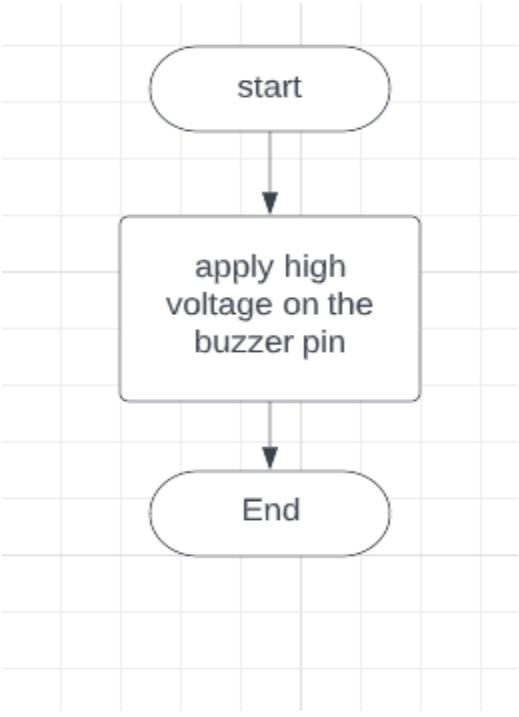


2.1.4 Buzzer

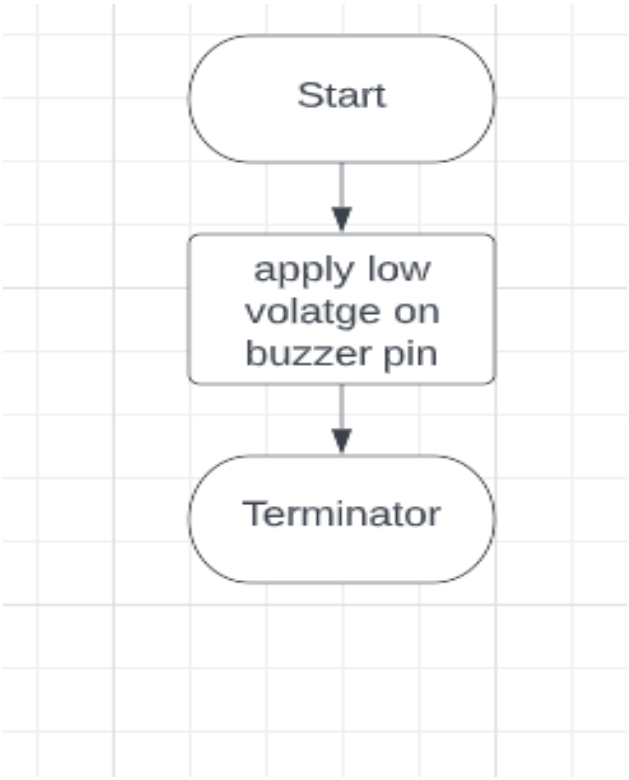
```
void BUZZ_init();
```




```
void BUZZ_on();
```

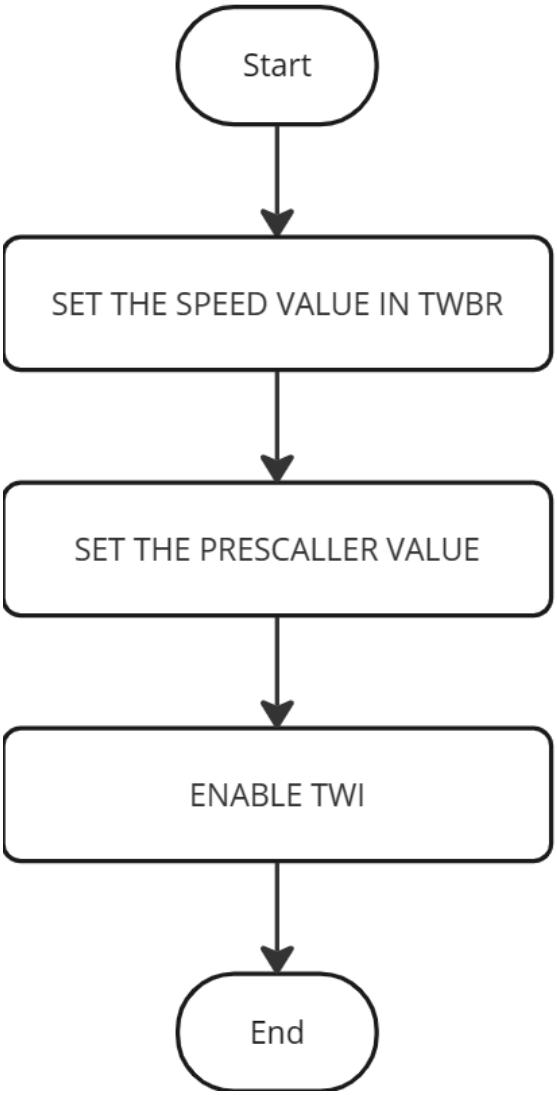


```
void BUZZ_off();
```

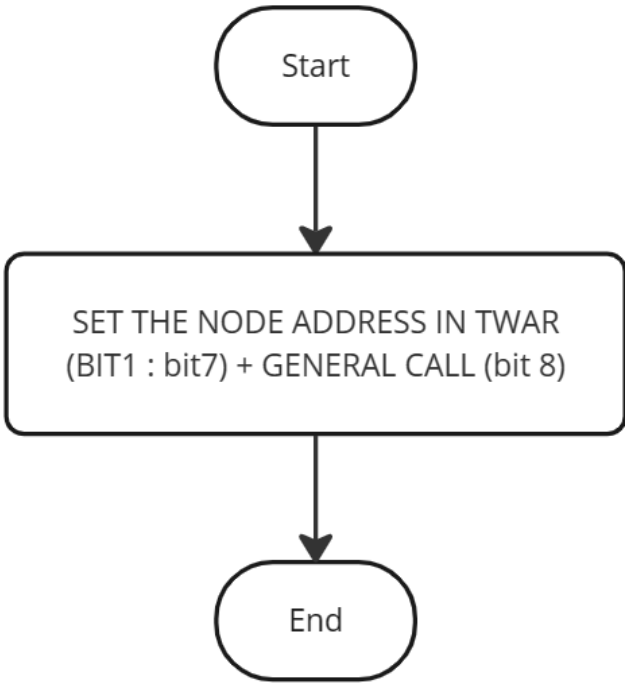


2.1.5 I2C

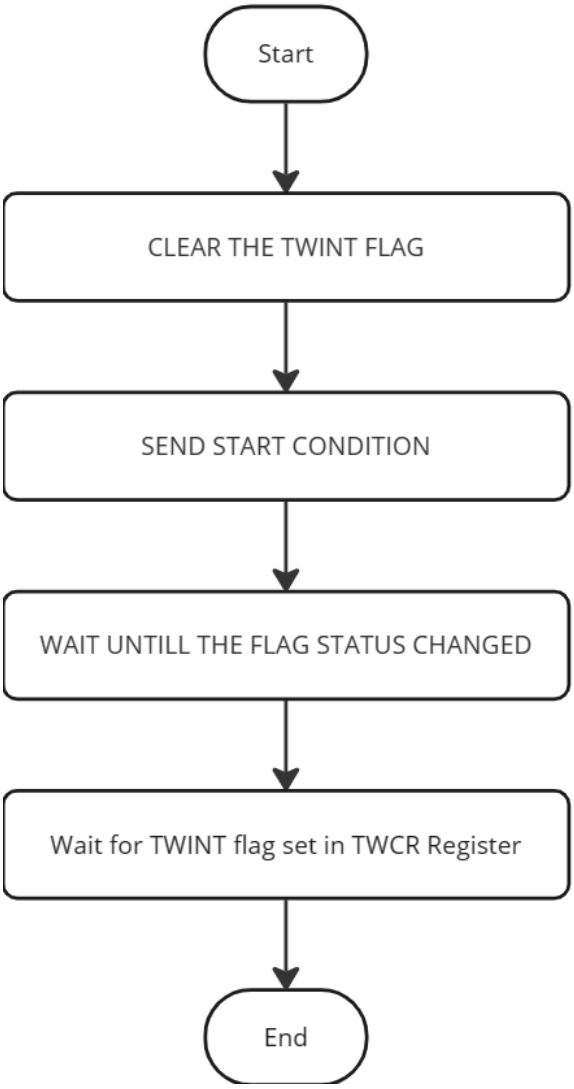
void TWI_init(void)



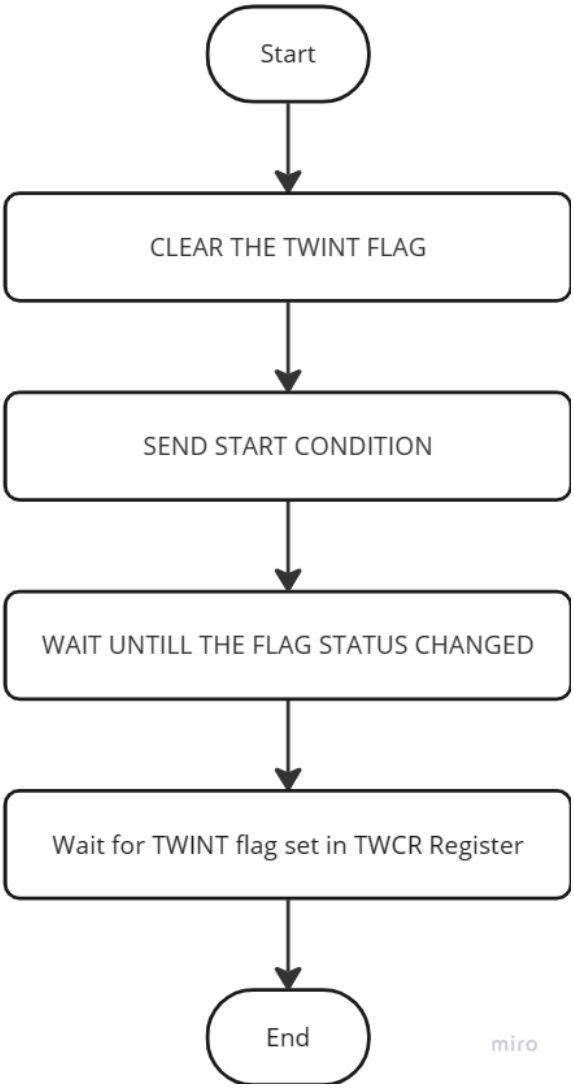
void TWI_setaddress(uint8_t u8_a_address)



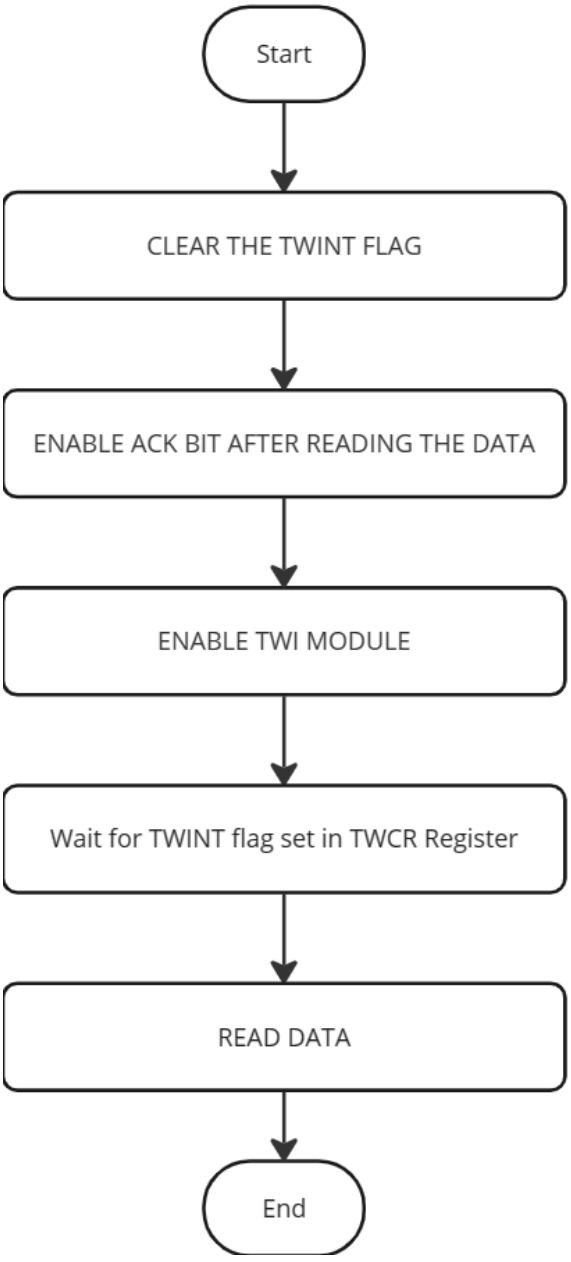
void TWI_start(void)



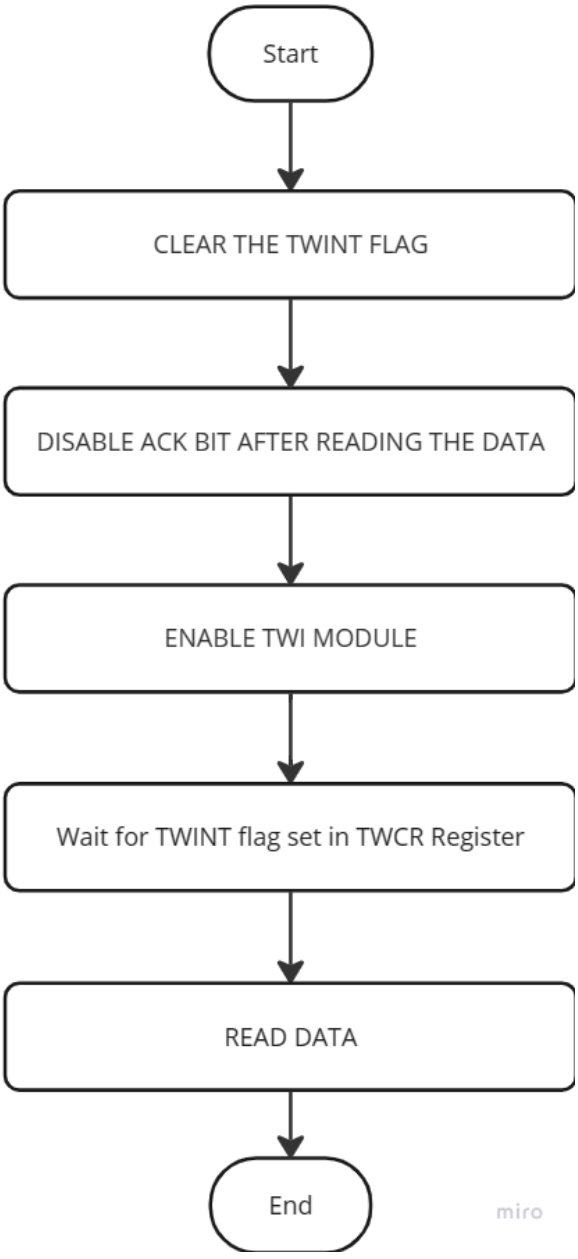
void TWI_repeatedstart(void)



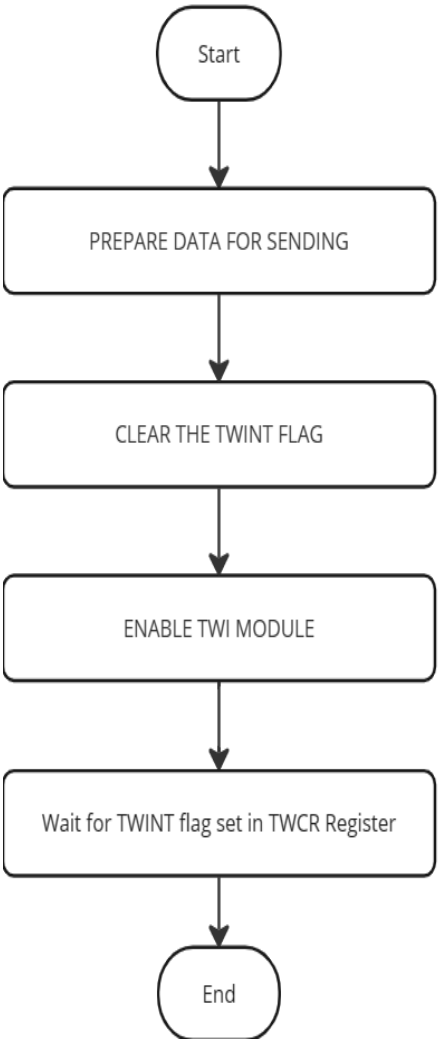
uint8_t TWI_readwithack(void)



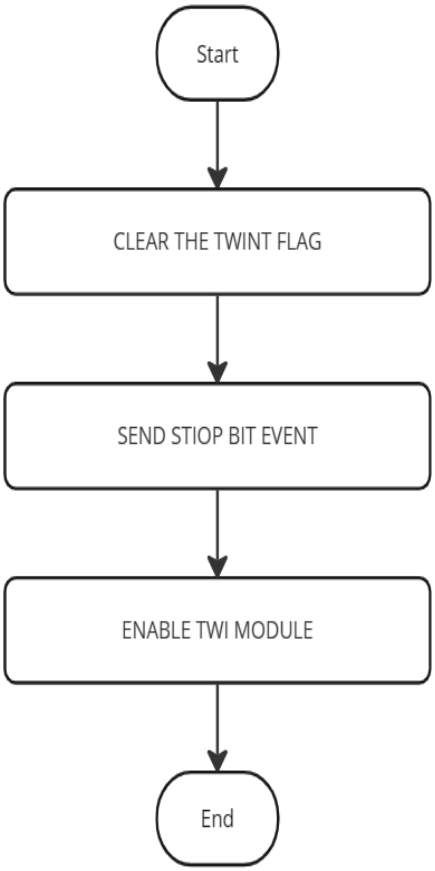
uint8_t TWI_readwithnack(void)



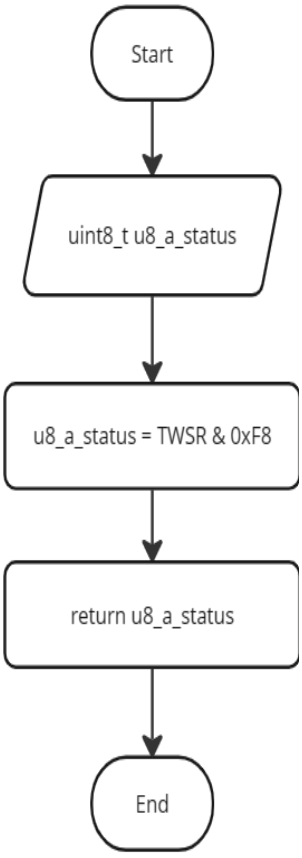
void TWI_write(uint8_t u8_a_data)



void TWI_stop(void)

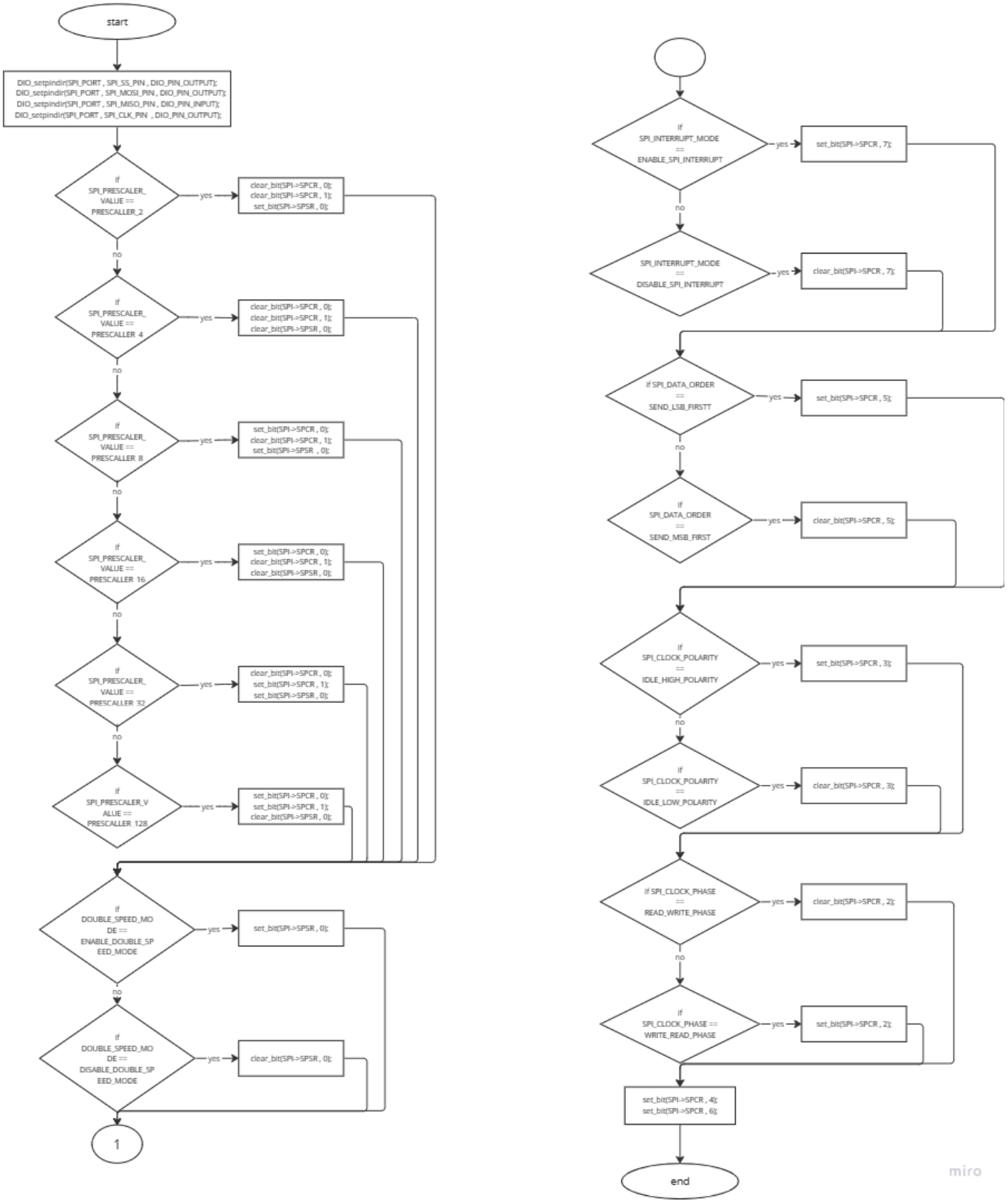


uint8_t TWI_getstatus(void)

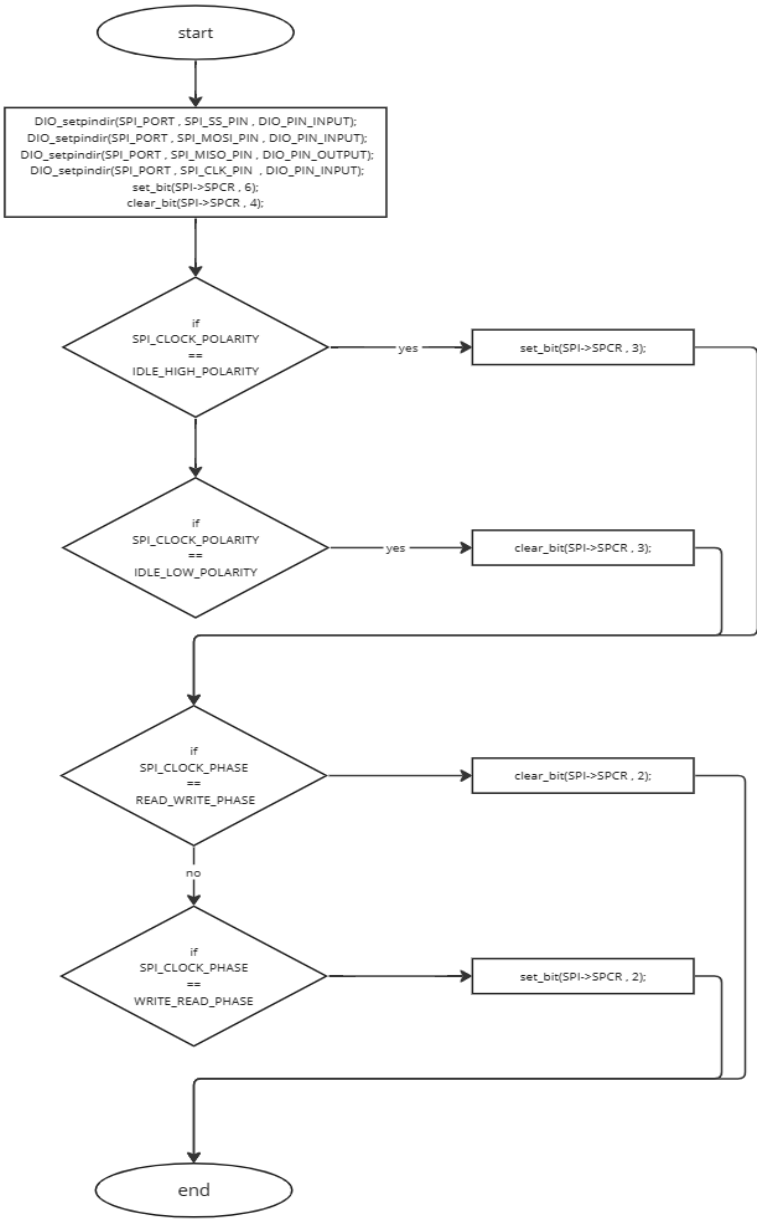


2.1.6 SPI

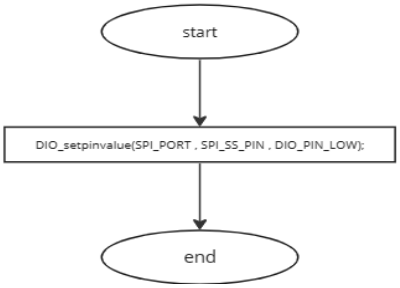
en_a_spierrstatus SPI_initmaster(void)



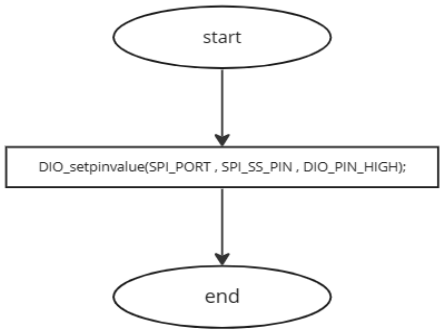
en_a_spierrstatus SPI_initslave(void)



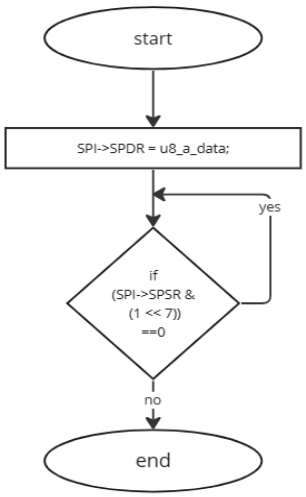
en_a_spierrstatus SPI_masterinittransmit(void)



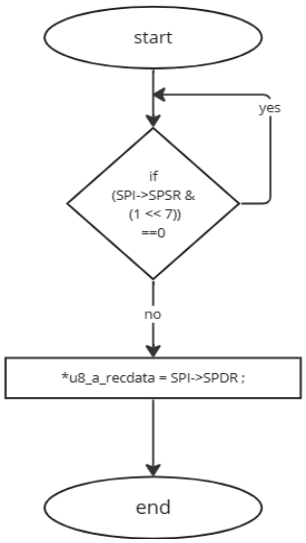
en_a_spierrstatus SPI_masterendtransmit(void)



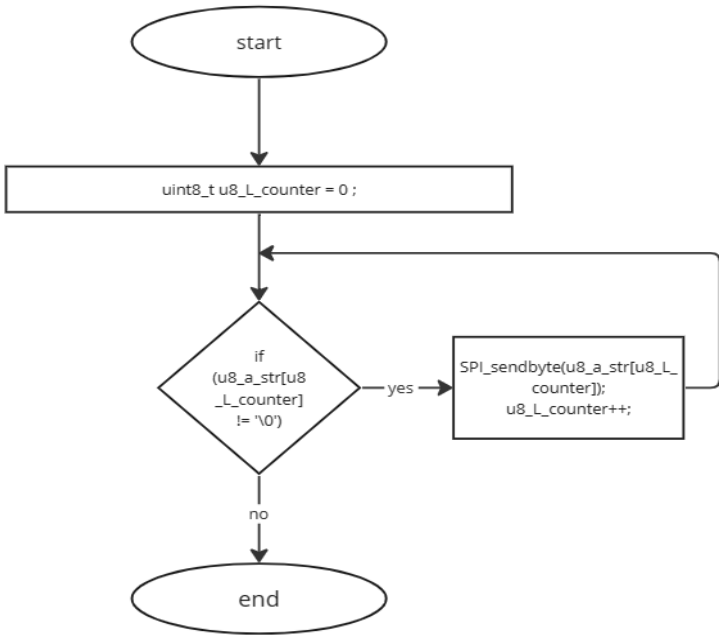
en_a_spierrstatus SPI_sendbyte(uint8_t u8_a_data)



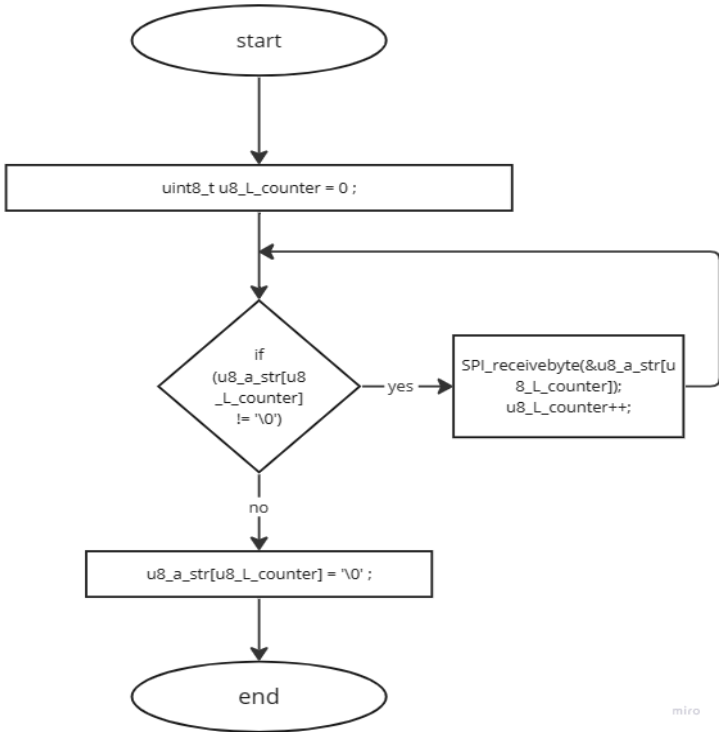
en_a_spierrstatus SPI_receivebyte(uint8_t * u8_a_recdata)



en_a_spierrstatus SPI_sendstring(const uint8_t * u8_a_str)

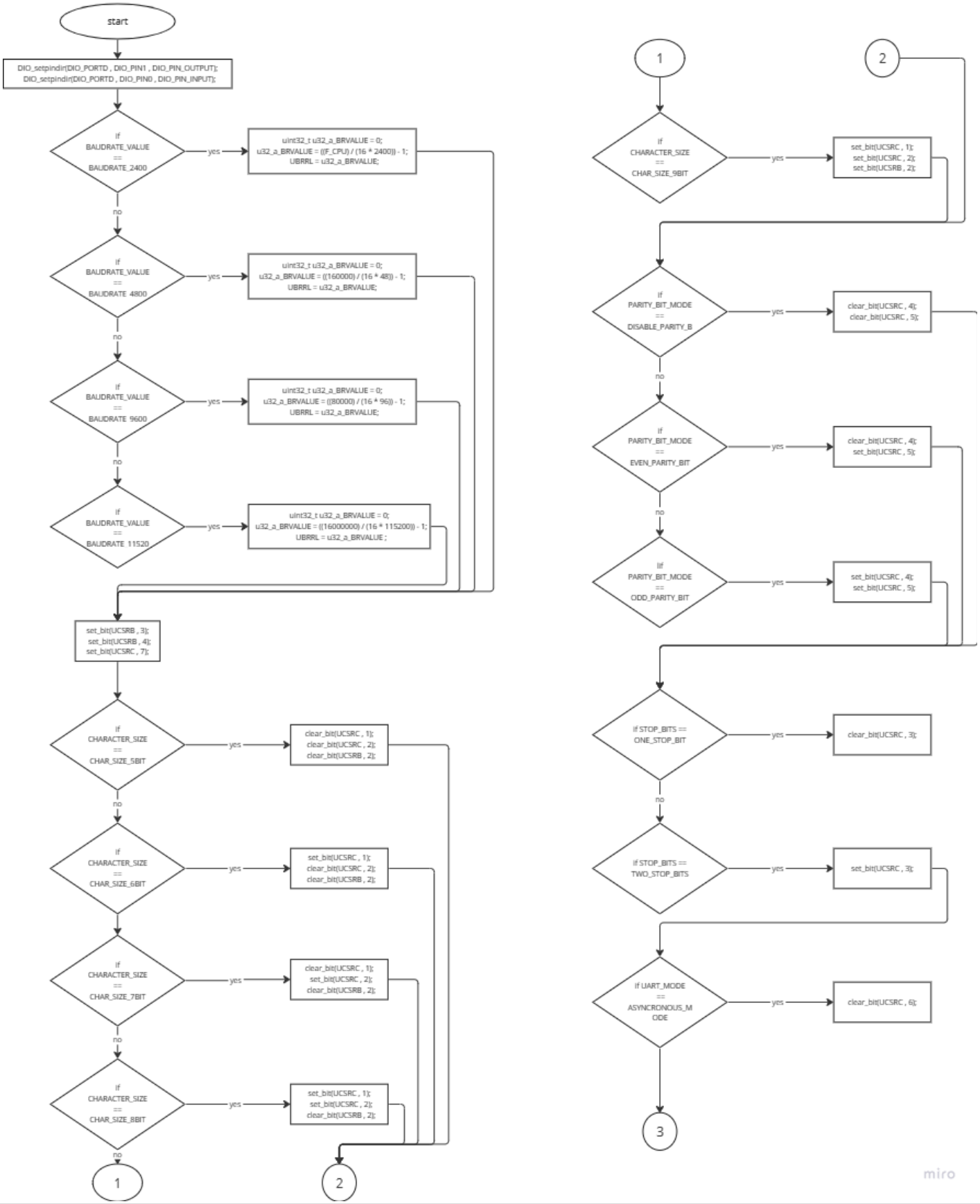


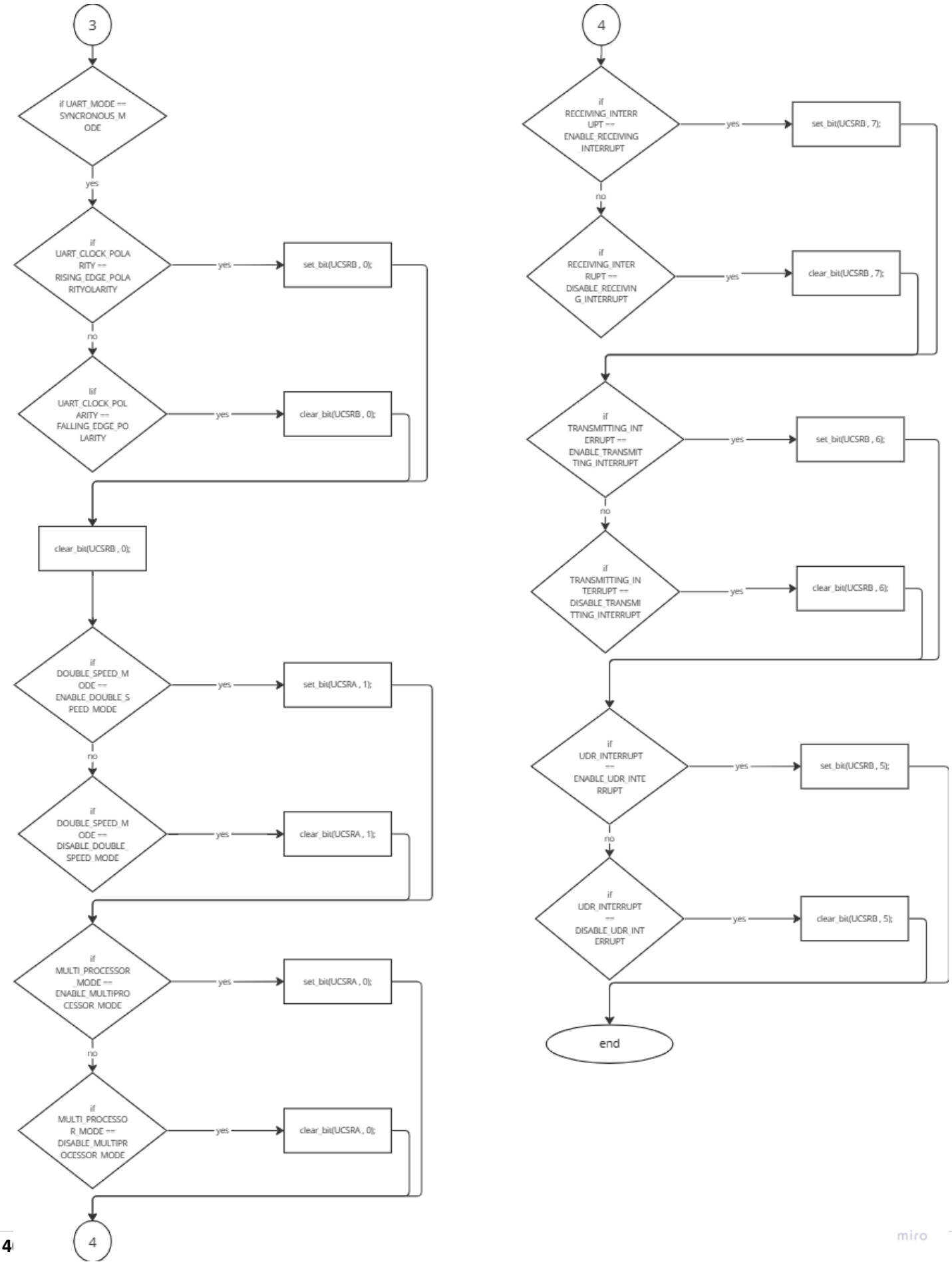
en_a_spierrstatus SPI_receivestring(uint8_t * u8_a_str)



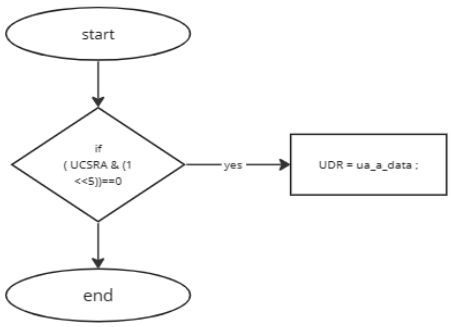
2.1.7 UART

uart_errorstatus UART_init(void)

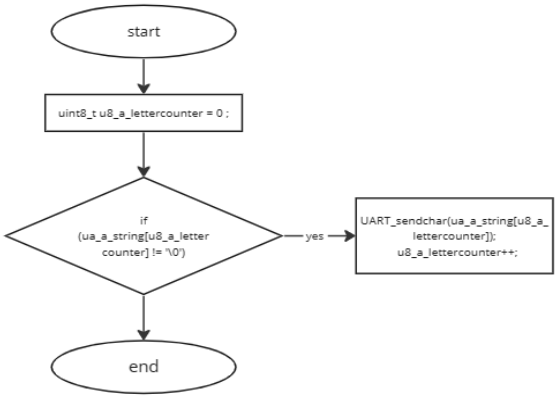



3
if UART_MODE == SYNCRONIOUS_MODE
yes
if UART_CLOCK_POLARITY == RISING_EDGE_POLARITYOLARITY
yes
set_bit(UCSRB , 0);
no
lif UART_CLOCK_POLARITY == FALLING_EDGE_POLARITY
yes
clear_bit(UCSRB , 0);
clear_bit(UCSRB , 0);
if DOUBLE_SPEED_MODE == ENABLE_DOUBLE_SPEED_MODE
yes
set_bit(UCSRA , 1);
no
if DOUBLE_SPEED_MODE == DISABLE_DOUBLE_SPEED_MODE
yes
clear_bit(UCSRA , 1);
if MULTI_PROCESSOR_MODE == ENABLE_MULTIPROCESSOR_MODE
yes
set_bit(UCSRA , 0);
no
if MULTI_PROCESSOR_MODE == DISABLE_MULTIPROCESSOR_MODE
yes
clear_bit(UCSRA , 0);
4
4
if RECEIVING_INTERRUPT == ENABLE_RECEIVING_INTERRUPT
yes
set_bit(UCSRB , 7);
no
if RECEIVING_INTERRUPT == DISABLE_RECEIVING_INTERRUPT
yes
clear_bit(UCSRB , 7);
if TRANSMITTING_INTERRUPT == ENABLE_TRANSMITTING_INTERRUPT
yes
set_bit(UCSRB , 6);
no
if TRANSMITTING_INTERRUPT == DISABLE_TRANSMITTING_INTERRUPT
yes
clear_bit(UCSRB , 6);
if UDR_INTERRUPT == ENABLE_UDR_INTERRUPT
yes
set_bit(UCSRB , 5);
no
if UDR_INTERRUPT == DISABLE_UDR_INTERRUPT
yes
clear_bit(UCSRB , 5);
end

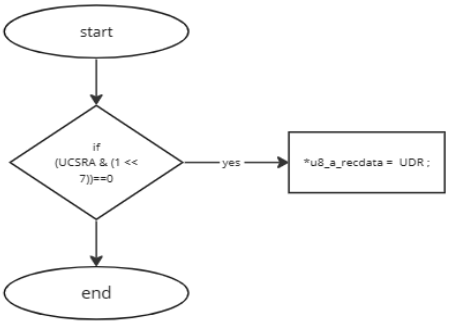
uart_errorstatus UART_sendchar(uint8_t ua_a_data)



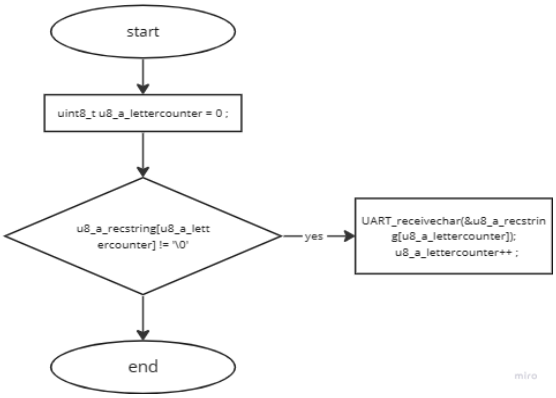
uart_errorstatus UART_sendstr(uint8_t * ua_a_string)



uart_errorstatus UART_receivechar(uint8_t * u8_a_recdata)

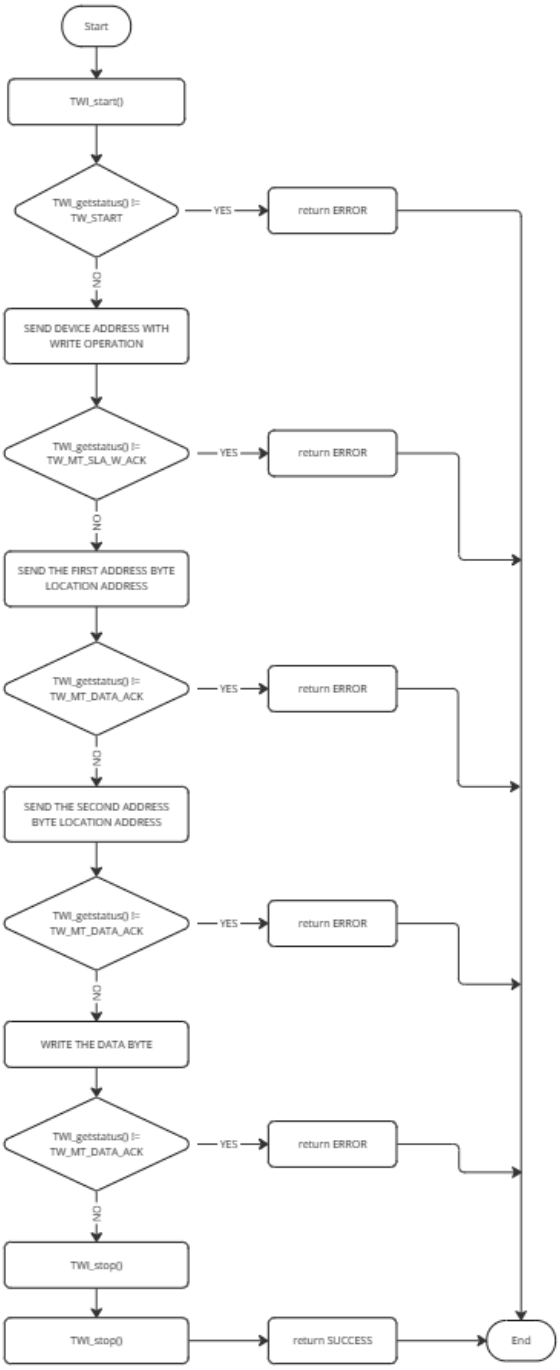


uart_errorstatus UART_receivestr(uint8_t * u8_a_recstring)

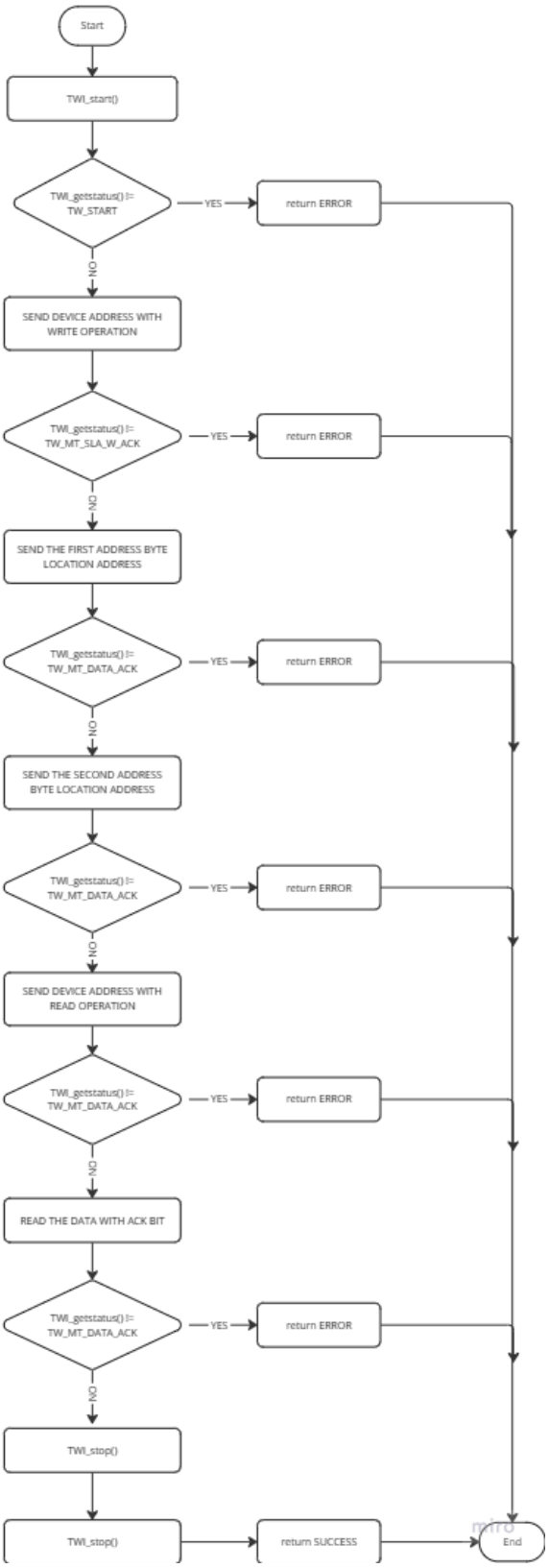


2.1.8 EEPROM

uint8_t EEPROM_writebyte(uint16_t u16_a_address , uint8_t u8_a_data , uint8_t u8_a_page_address)

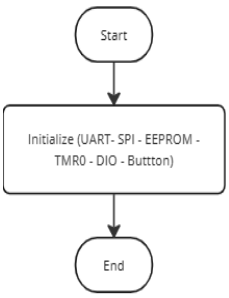


uint8_t EEPROM_readbyte(uint16_t u16_a_address , uint8_t *u8_a_data , uint8_t u8_a_page_address)

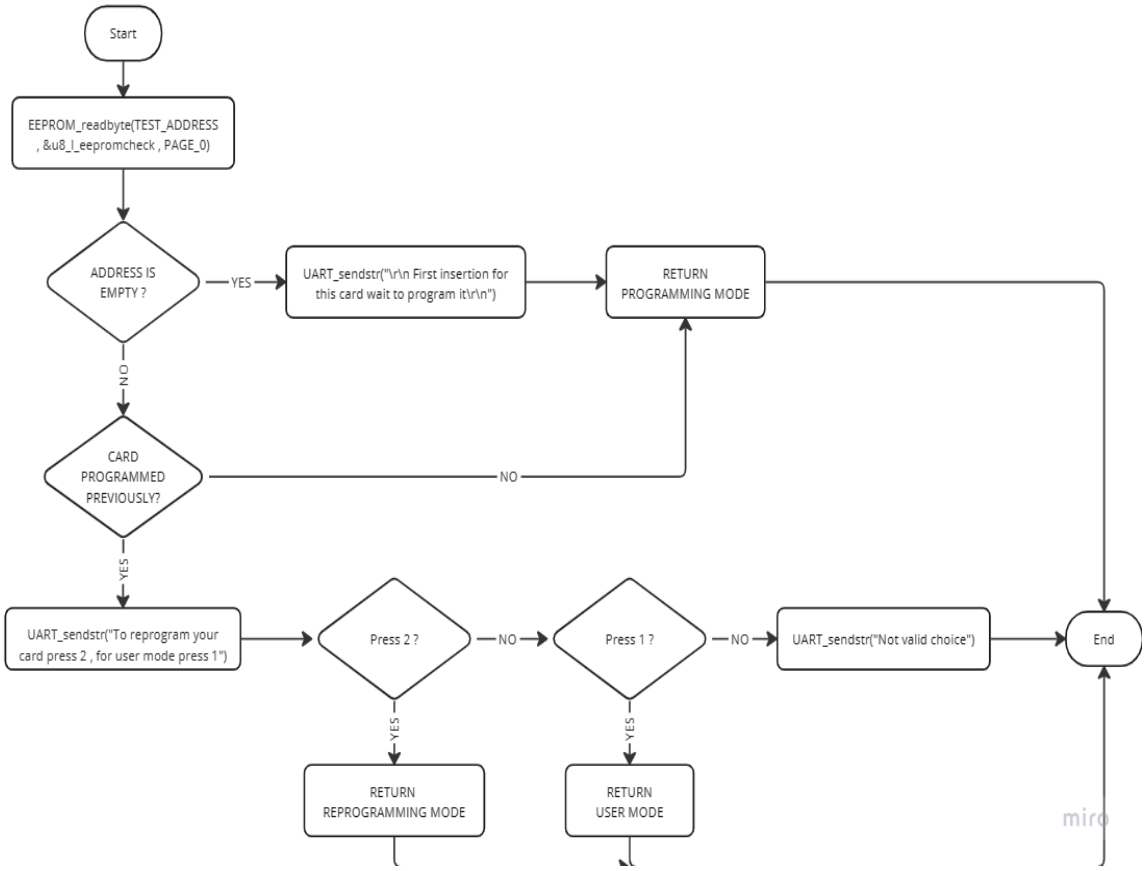


2.1.9 App_Card

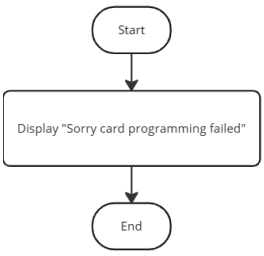
void APP_init()



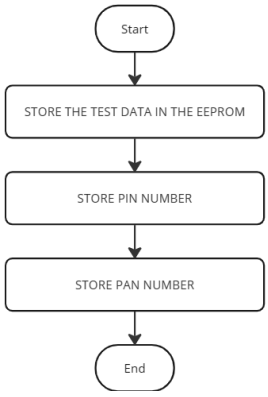
uint8_t APP_entrypoint()



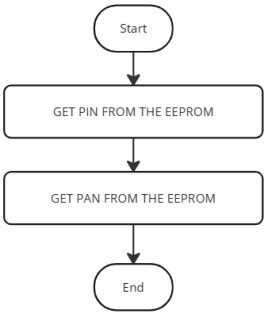
void APP_cardfailed()



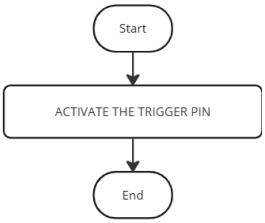
void APP_storecard()

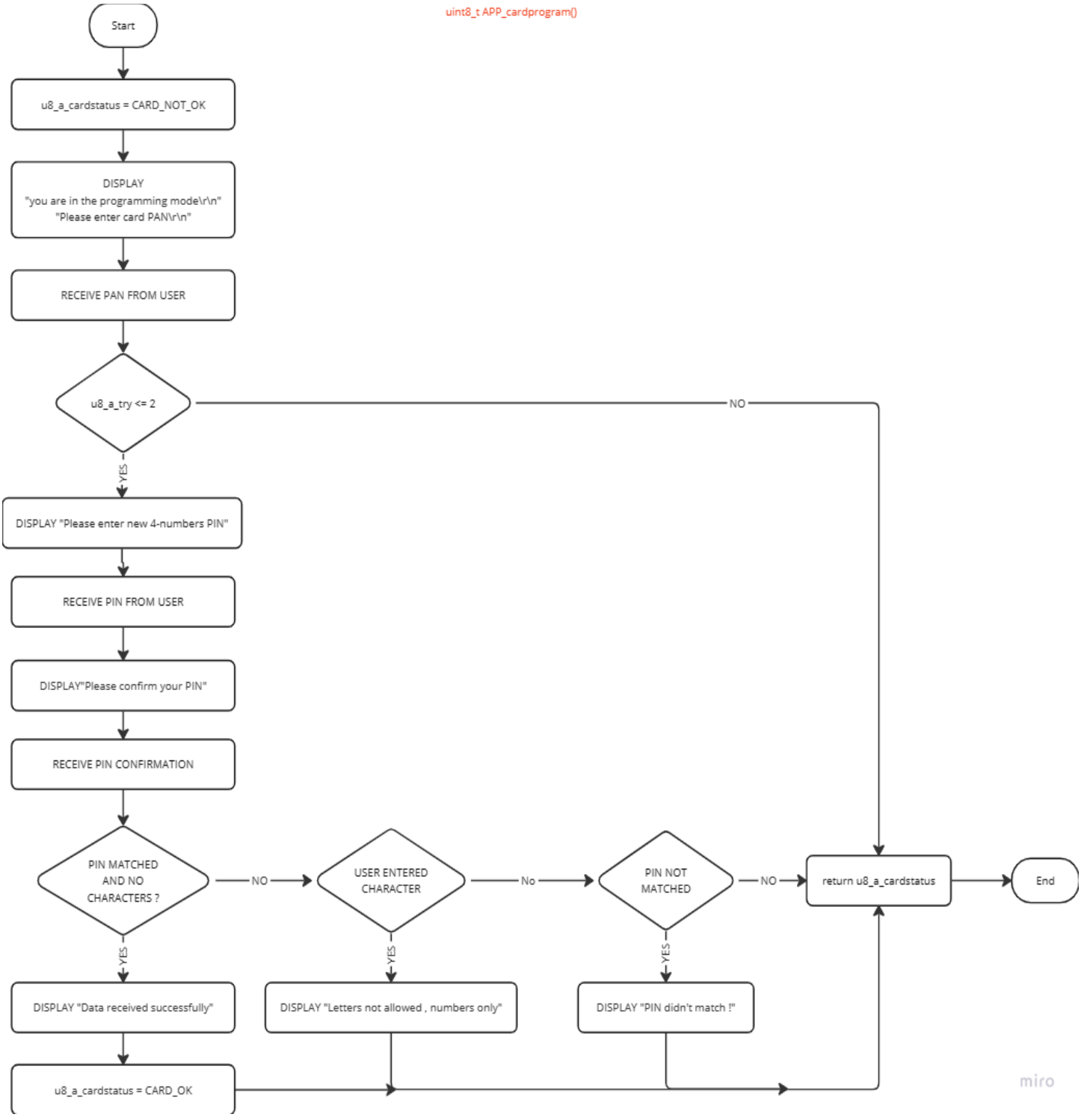


void APP_getcarddata(void)

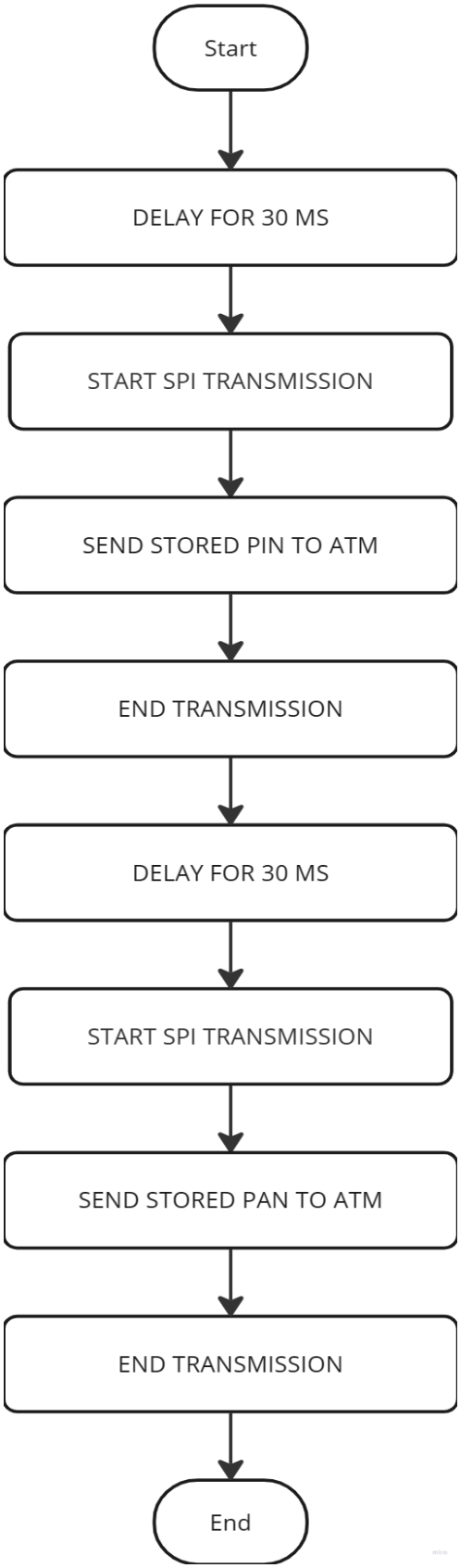


APP_sendtrigger()



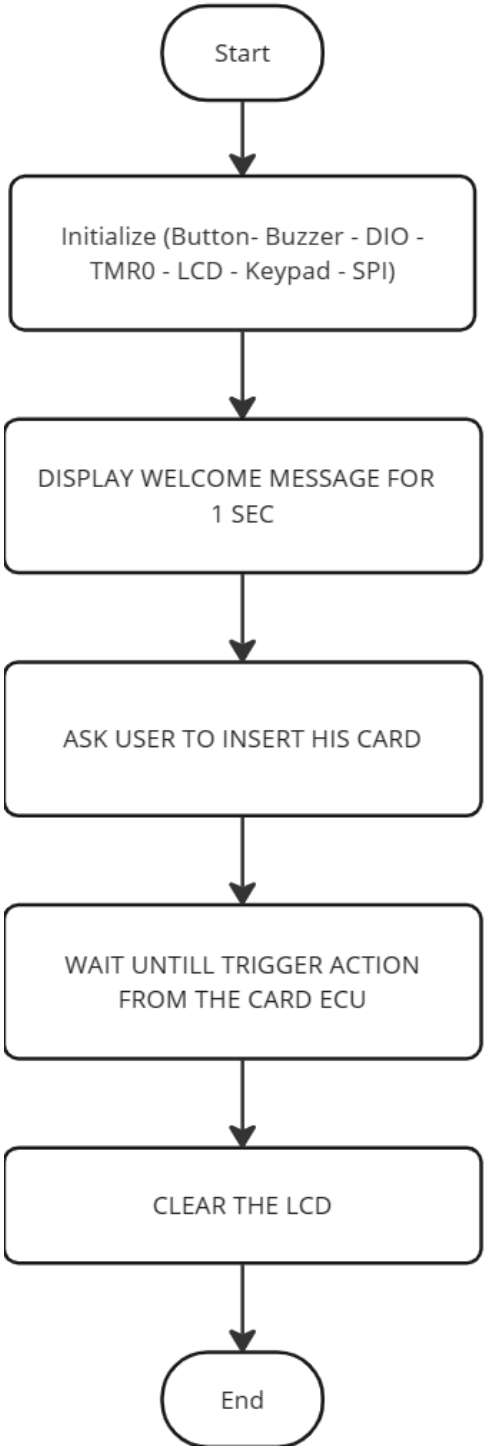

uint8_t APP_cardprogram()
Start
u8_a_cardstatus = CARD_NOT_OK
DISPLAY
"you are in the programming mode\r\n"
"Please enter card PAN\r\n"
RECEIVE PAN FROM USER
u8_a_try <= 2
NO
YES
DISPLAY "Please enter new 4-numbers PIN"
RECEIVE PIN FROM USER
DISPLAY"Please confirm your PIN"
RECEIVE PIN CONFIRMATION
PIN MATCHED AND NO CHARACTERS ?
NO
USER ENTERED CHARACTER
No
PIN NOT MATCHED
NO
return u8_a_cardstatus
End
YES
YES
YES
DISPLAY "Data received successfully"
DISPLAY "Letters not allowed , numbers only"
DISPLAY "PIN didn't match !"
u8_a_cardstatus = CARD_OK
miro

oid APP_sendcarddata()

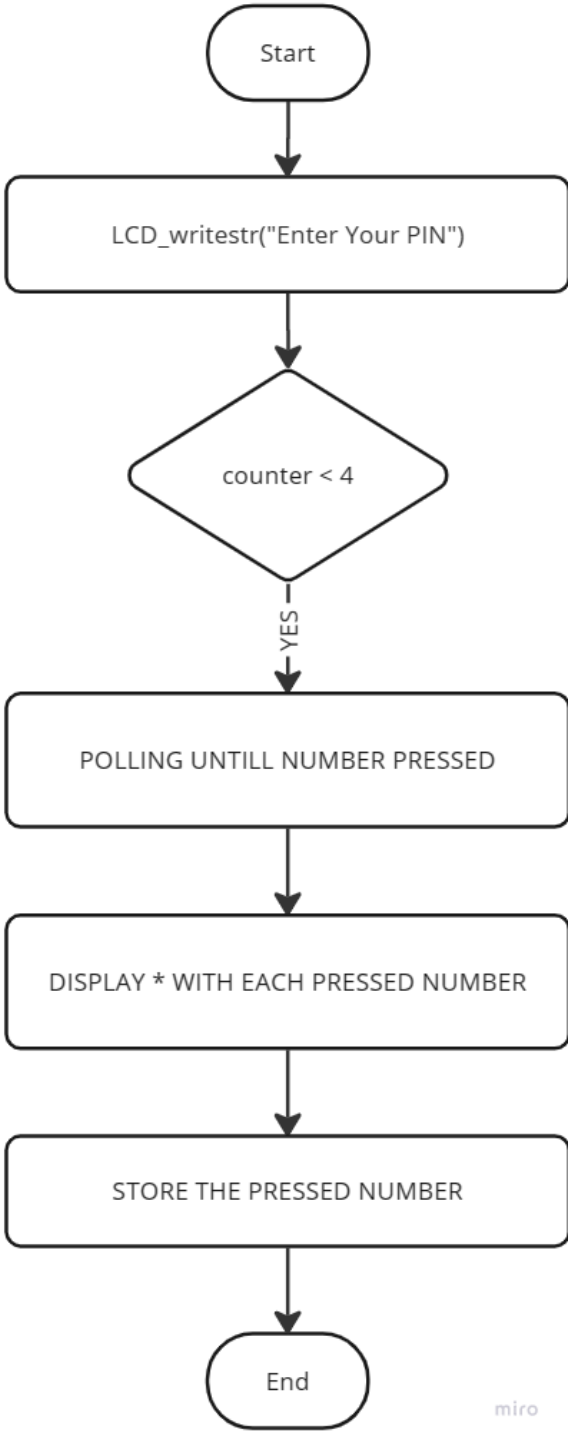


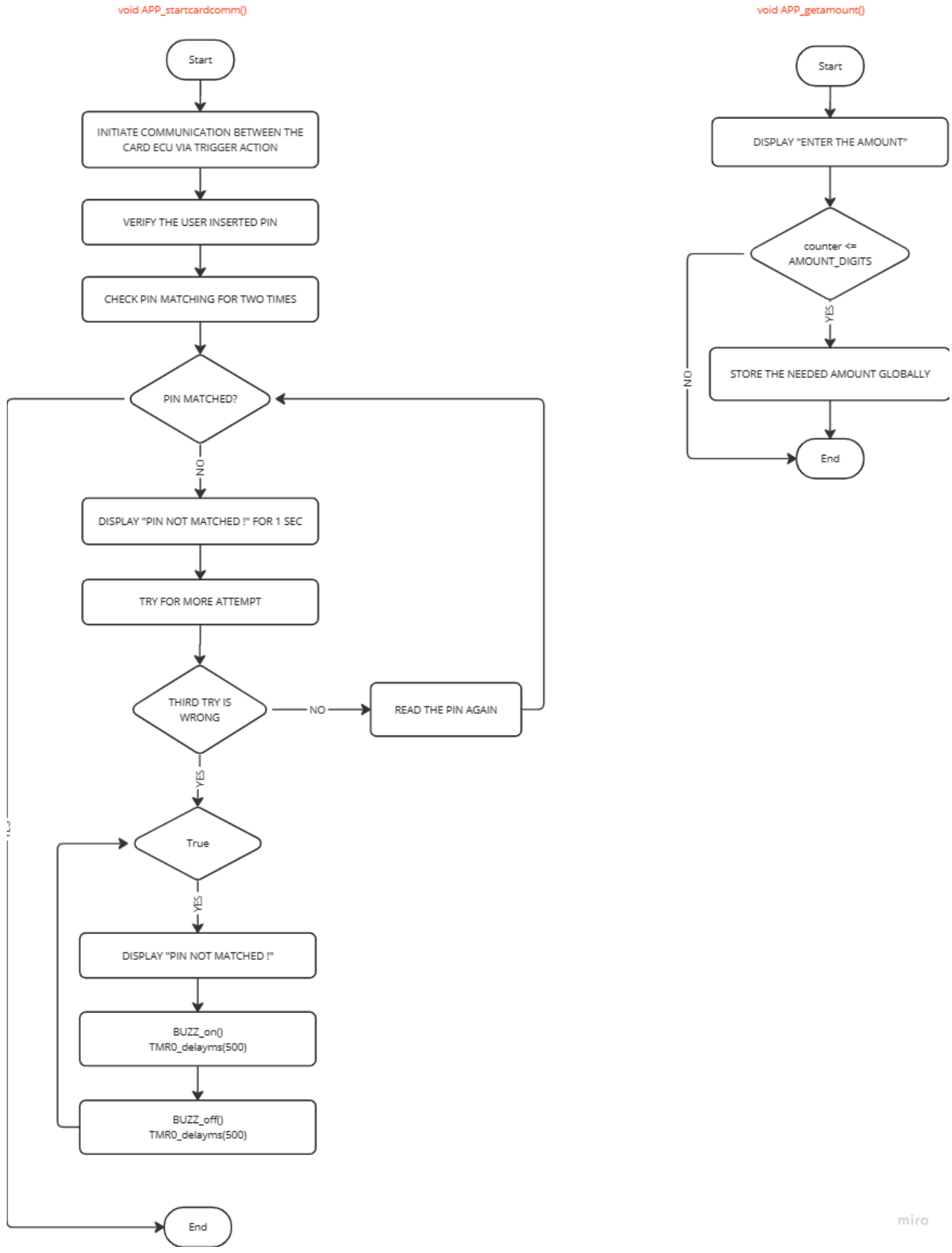
2.1.10 App_ATM

void APP_init()



void APP_readuserpin()




void APP_startcardcomm()
Start
INITIATE COMMUNICATION BETWEEN THE CARD ECU VIA TRIGGER ACTION
VERIFY THE USER INSERTED PIN
CHECK PIN MATCHING FOR TWO TIMES
PIN MATCHED?
NO
DISPLAY "PIN NOT MATCHED !" FOR 1 SEC
TRY FOR MORE ATTEMPT
THIRD TRY IS WRONG
NO
READ THE PIN AGAIN
YES
True
YES
DISPLAY "PIN NOT MATCHED !"
BUZZ_on()
TMR0_delayms(500)
BUZZ_off()
TMR0_delayms(500)
End
void APP_getamount()
Start
DISPLAY "ENTER THE AMOUNT"
counter <= AMOUNT_DIGITS
YES
STORE THE NEEDED AMOUNT GLOBALLY
NO
End
miro

void APP_cardvalidate(void)