Design document

project title: LED sequence V2.0

Name: Hazem Ashraf

Project Description

LED sequence project consists of

- ***Hardware components***
  - *Four LEDs (LED0, LED1, LED2, LED3)*
  - One button (BUTTON0)
- ***Software Requirements***

Initially, all LEDs are OFF

Once **BUTTON1** is pressed, **LED0** will be **ON**

Each press further will make another LED is **ON**

At the **fifth press**, **LED0** will changed to be **OFF**

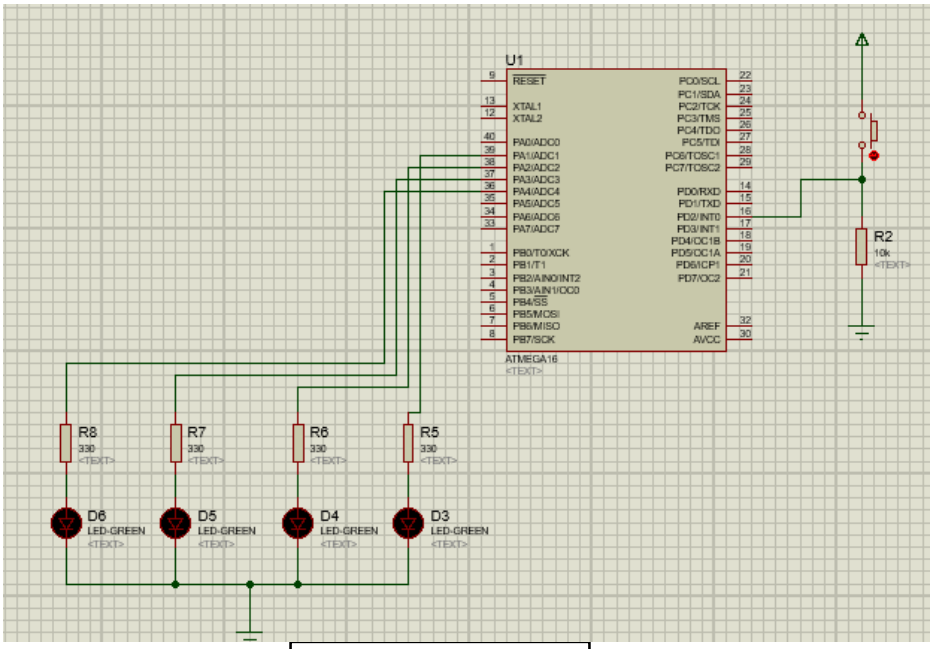
Each **press further** will make only one LED is **OFF**

This will be repeated forever

The sequence is described below

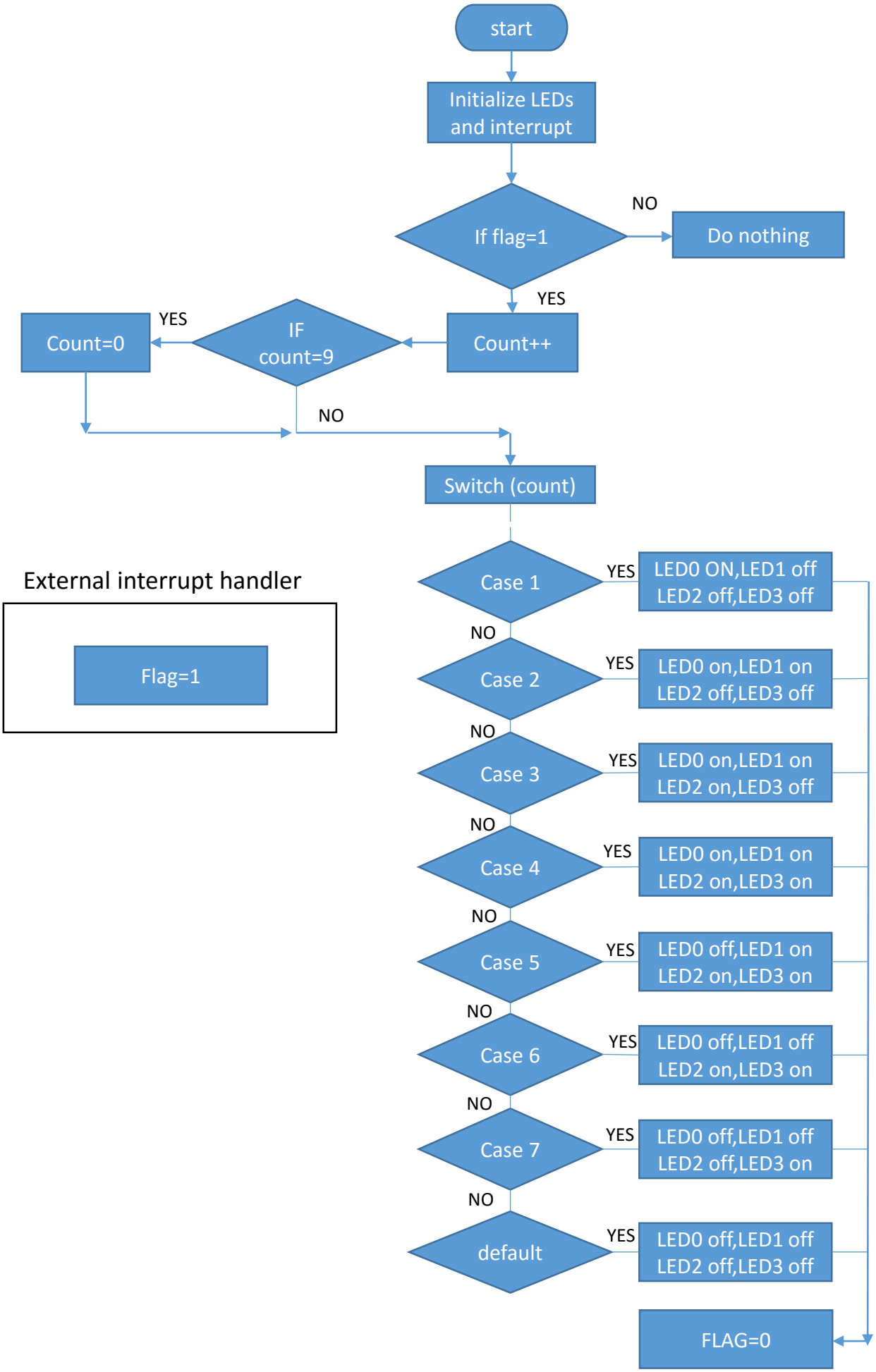
- Initially (OFF, OFF, OFF, OFF)
- Press 1 (ON, OFF, OFF, OFF)
- Press 2 (ON, ON, OFF, OFF)
- Press 3 (ON, ON, ON, OFF)
- Press 4 (ON, ON, ON, ON)
- Press 5 (OFF, ON, ON, ON)
- Press 6 (OFF, OFF, ON, ON)
- Press 7 (OFF, OFF, OFF, ON)
- Press 8 (OFF, OFF, OFF, OFF)
- Press 9 (ON, OFF, OFF, OFF)

USE EXTERNAL INTERRUPTS

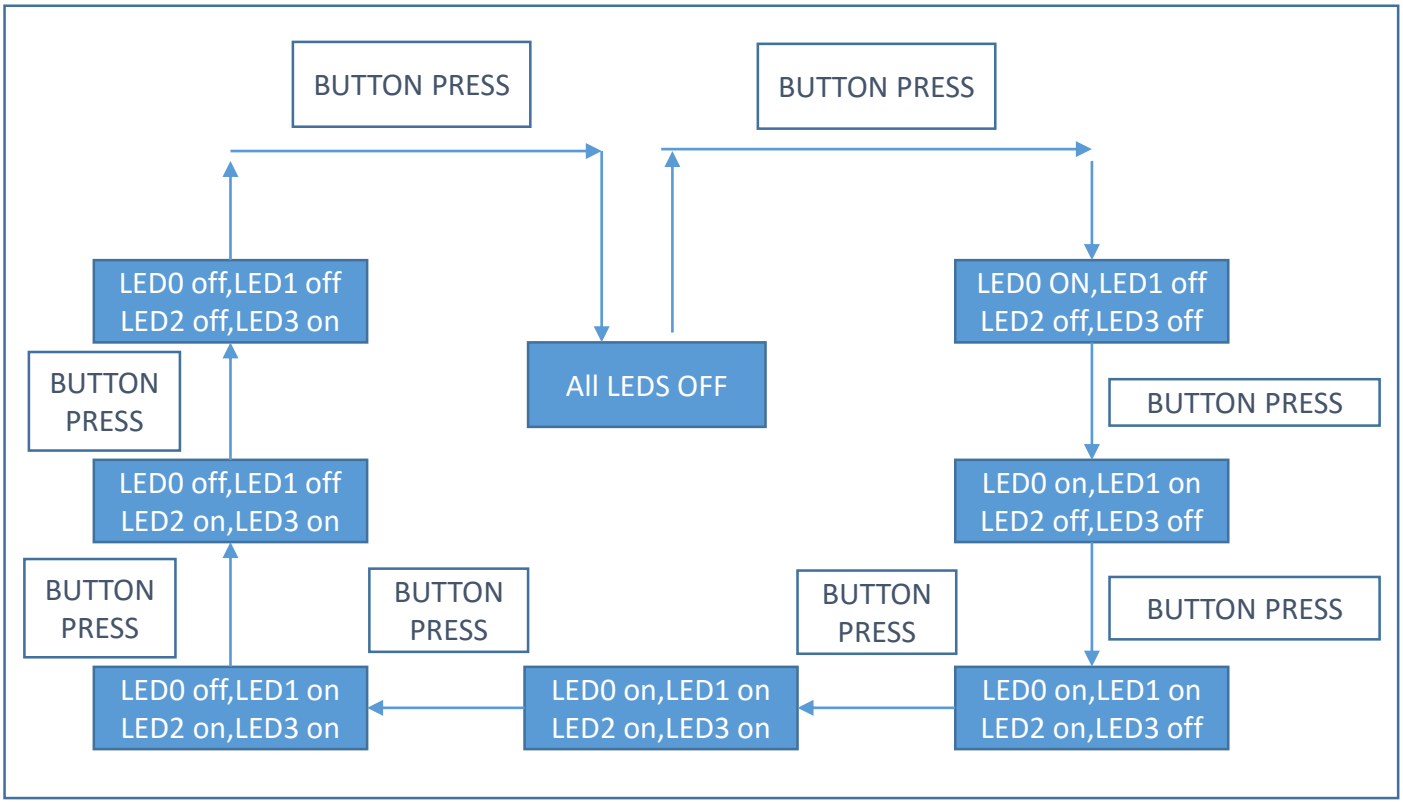


Circuit wiring

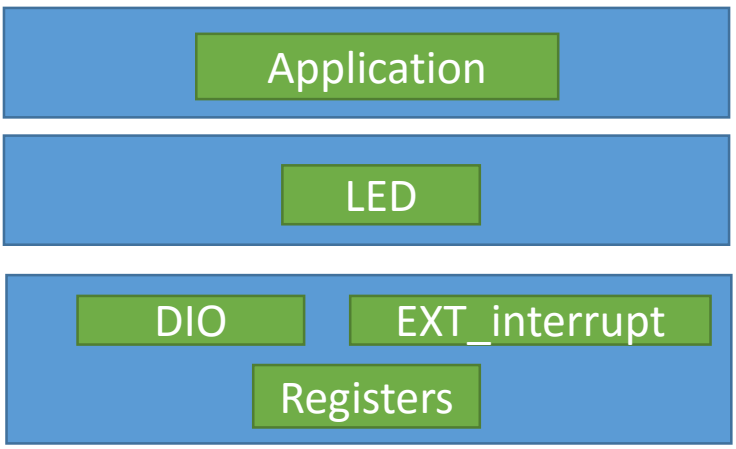
Project flowchart diagram



Project state machine diagram



Layered architecture



Project Modules APIs

- GPIO module APIs

/*============ TYPE DEFINITION ============*/

```
typedef enum{
PIN_INPUT,PIN_OUTPUT
}EN_PIN_DIRECTION;

typedef enum
{
PORT_INPUT,PORT_OUTPUT=0xFF
}EN_PORT_DIRECTION;

typedef enum{
Low,High
}EN_PIN_VALUE;

typedef enum{
LOW,HIGH=0xFF
}EN_PORT_VALUE;

typedef enum{
FAILED,SUCCESS
}EN_STATE;

typedef struct{
uint8 pinx;
uint8 ddrx;
uint8 portx;
}ST_register_name;
typedef ST_register_name* REG_NAME;
```

/*============ FUNCTION PROTOTYPE============*/

EN_STATE **pinMode(uint8 pin_no,EN_PIN_DIRECTION pin_direction);**

Description:

- PinMode:used to set pin direction input/output
- function parameters
- pin_no:pin number to set
- pin_direction:direction of the pin
- Return success pin number is in the range, FAILED if pin number out of the range

EN_STATE **digitalWrite(uint8 pin_no,EN_PIN_VALUE pin_val);**

Description:

- digitalWrite:used to write high/low to specific pin
- function parameters
- pin_no:pin number to write
- pin_val:output value high / low
- Return success pin number is in the range, FAILED if pin number out of the range

Project Modules APIs

- GPIO module APIs

EN_STATE **digitalRead(uint8 pin_no,uint8 *pin_val);**

Description:

- digitalRead:used to read specific pin value
- function parameters
- pin_no:pin number to read
- pin_val:address to variable of the return reading
- Return success pin number is in the range, FAILED if pin number out of the range

EN_STATE portMode(REG_NAME port,EN_PORT_DIRECTION port_direction);

Description:

- portMode: used to specific port direction
- function parameters
- port: port name (PORTA-PORTB-PORTC-PORTD)
- port_direction: direction of the port
- Return success port name is in the range, FAILED if port name out of the range

EN_STATE digitalWrite_Port(REG_NAME port,EN_PORT_VALUE port_val);

Description

- digitalWrite_PORT:used to write high/low to specific port
- function parameters
- port: port name (PORTA-PORTB-PORTC-PORTD)
- port_val: output value HIGH / LOW
- Return success port name is in the range, FAILED if port name out of the range

EN_STATE digitalRead_Port(REG_NAME port,uint8 *port_val);

Description

- digitalRead_PORT:used to read specific port value
- function parameters
- port: port name (PORTA-PORTB-PORTC-PORTD)
- port_val: address to variable of the return reading
- Return success port name is in the range, FAILED if port name out of the range

EN_STATE **Enable_PULLUP (uint8 pin_no);**

Description

- active internal pull up resistor for specific pin
- pin_no:pin number to set
- Return success pin number is in the range, FAILED if pin number out of the range

Project Modules APIs

- EXT-interrupt module APIs

/*============= TYPE DEFINITION =============*/

typedef enum{
EN_INT0,EN_INT1,EN_INT2
}EN_INT_source;

typedef enum{
LOW_LEVEL,ANY_CHANGE,FALLING,RISING
}EN_INT_TRIGGER;

typedef enum{
INT_FAILED,INT_SUCCESS
}EN_INT_error;

typedef struct{
EN_INT_source source;
EN_INT_TRIGGER trigger;
}ST_INT_Config;

#define INT0_pin 2 //PD2
#define INT1_pin 3 //PD3
#define INT2_pin 3 //PB2

/*============= FUNCTION PROTOTYPE =============*/

EN_INT_error **INT_init(ST_INT_Config* Int_config)**

Description

- INT_init: used to initialize the interrupt by:
- disable global interrupt
- enable external interrupt source and set pin to input
- set external interrupt trigger signal type
- enable global interrupt
- Function parameters
- Int_config: pointer to structure of ST_INT_Config
- Return : FAILED if passing parameters is not correct, SUCCESS if the passing parameters is correct

void INT0_setCallBack(void(*a_ptr)(void));

Description:

INT0_setCallBack:used to set call back function for external INT_0

void INT1_setCallBack(void(*a_ptr)(void));

Description:

INT1_setCallBack:used to set call back function for external INT_1

Project Modules APIs

- EXT-interrupt module APIs

void INT2_setCallBack(void(*a_ptr)(void));

Description:

INT2_setCallBack:used to set call back function for external INT_2

void INT_Deinit(ST_INT_Config* Int_config);

Description

- INT_init: used to initialize the interrupt by:
- Disable specific external interrupt source

Project Modules APIs

❑ LED module APIs

/*============= MACRO DEFINITION =============*/

#define LED_logic 1 //1:positive logic , 2:negative logic

Description : Macro used to configure LED logic connection

/*============= FUNCTION PROTOTYPE =============*/

EN_STATE **LED_init(uint8 led);**

Description:

- LED_init: used to initialize LED direction and initial value for the pin
- function parameters
- Led: pin number to be set
- Return success pin number is in the range, FAILED if pin number out of the range

EN_STATE **LED_digitalwrite(uint8 led,EN_PIN_VALUE value);**

Description

- LED_digitalwrite**:** used to write high/low to specific led
- function parameters
- Led: pin number to be set
- Value: led high/low
- Return success pin number is in the range, FAILED if pin number out of the range

❑ BUTTON module APIs

/*============= TYPE DEFINITION =============*/

typedef enum{

disable,enable

}EN_internal_pullup;

/*============= FUNCTION PROTOTYPE =============*/

EN_STATE Button_init(uint8 pin,EN_internal_pullup state);

Description

- Button_init: used to initialize BUTTON direction and set internal pullup resistor
- function parameters
- pin: pin number to be set
- State: to disable/enable internal pullup resistor
- Return success pin number is in the range, FAILED if pin number out of the range

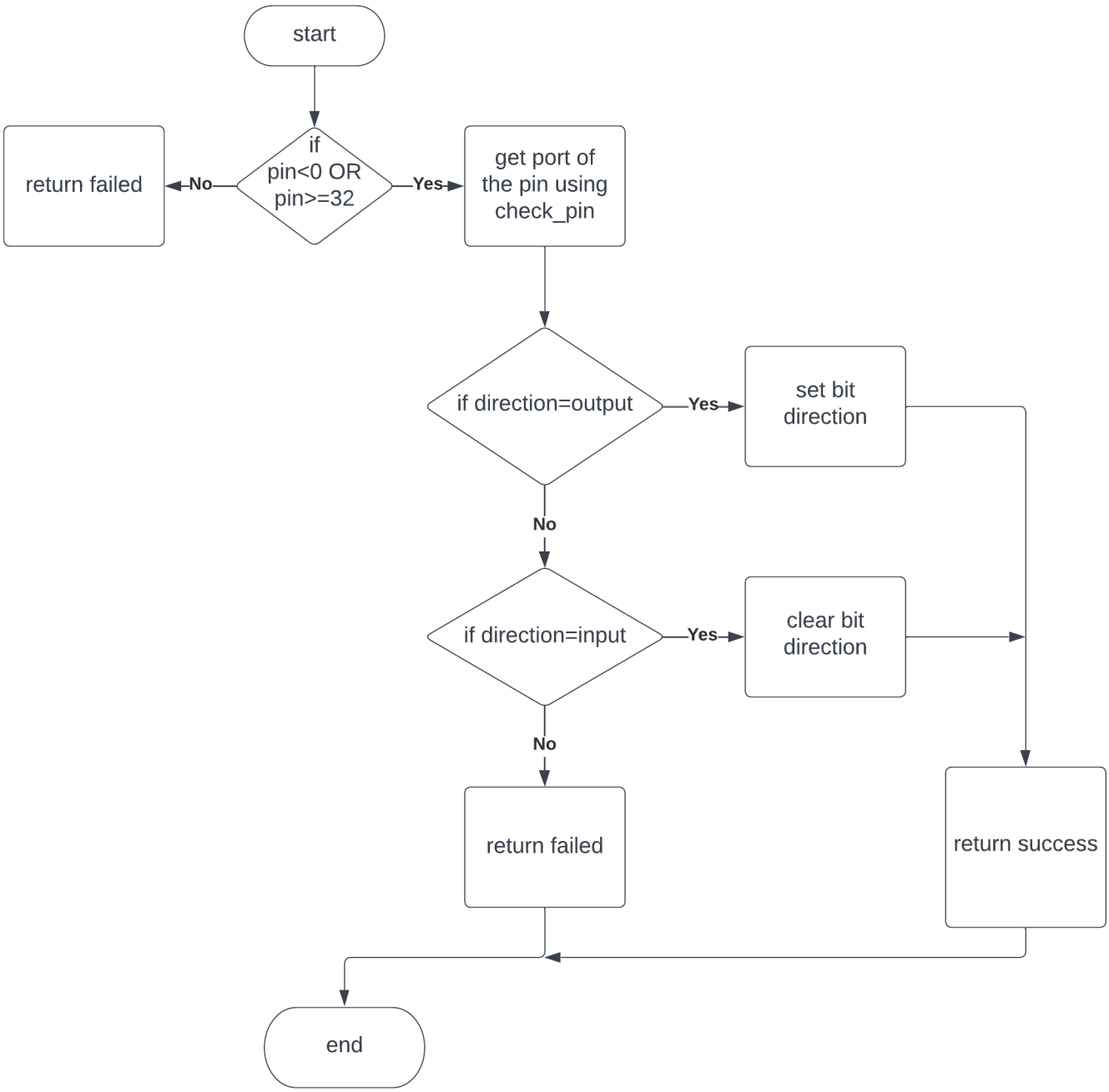
uint8 Button_Read(uint8 pin);

Description

- Button_Read: used to read button state high/low
- function parameters
- pin: pin number to read
- Return button state high / low

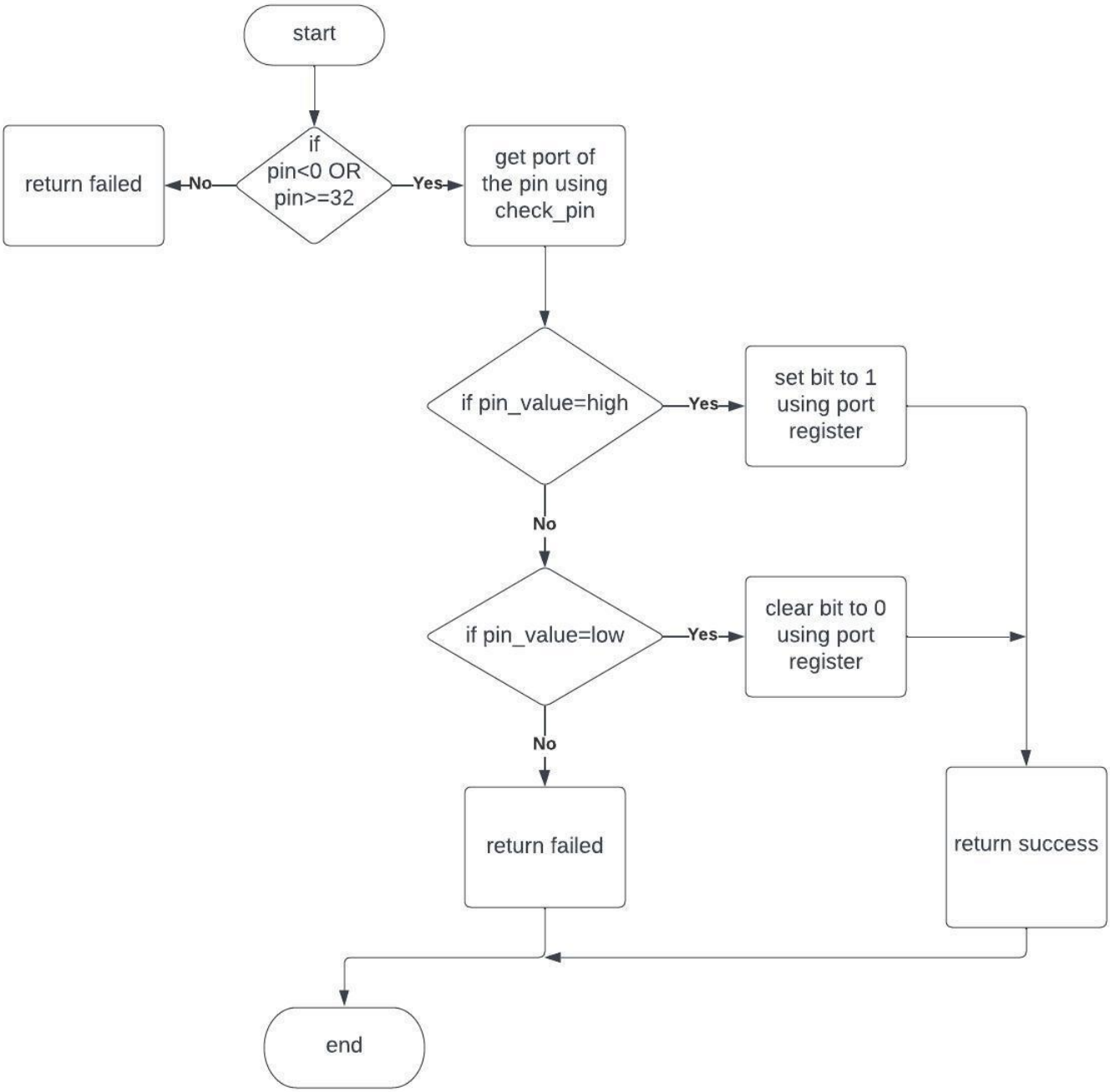
APIs flowcharts

EN_STATE **pinMode(uint8 pin_no,EN_PIN_DIRECTION pin_direction);**



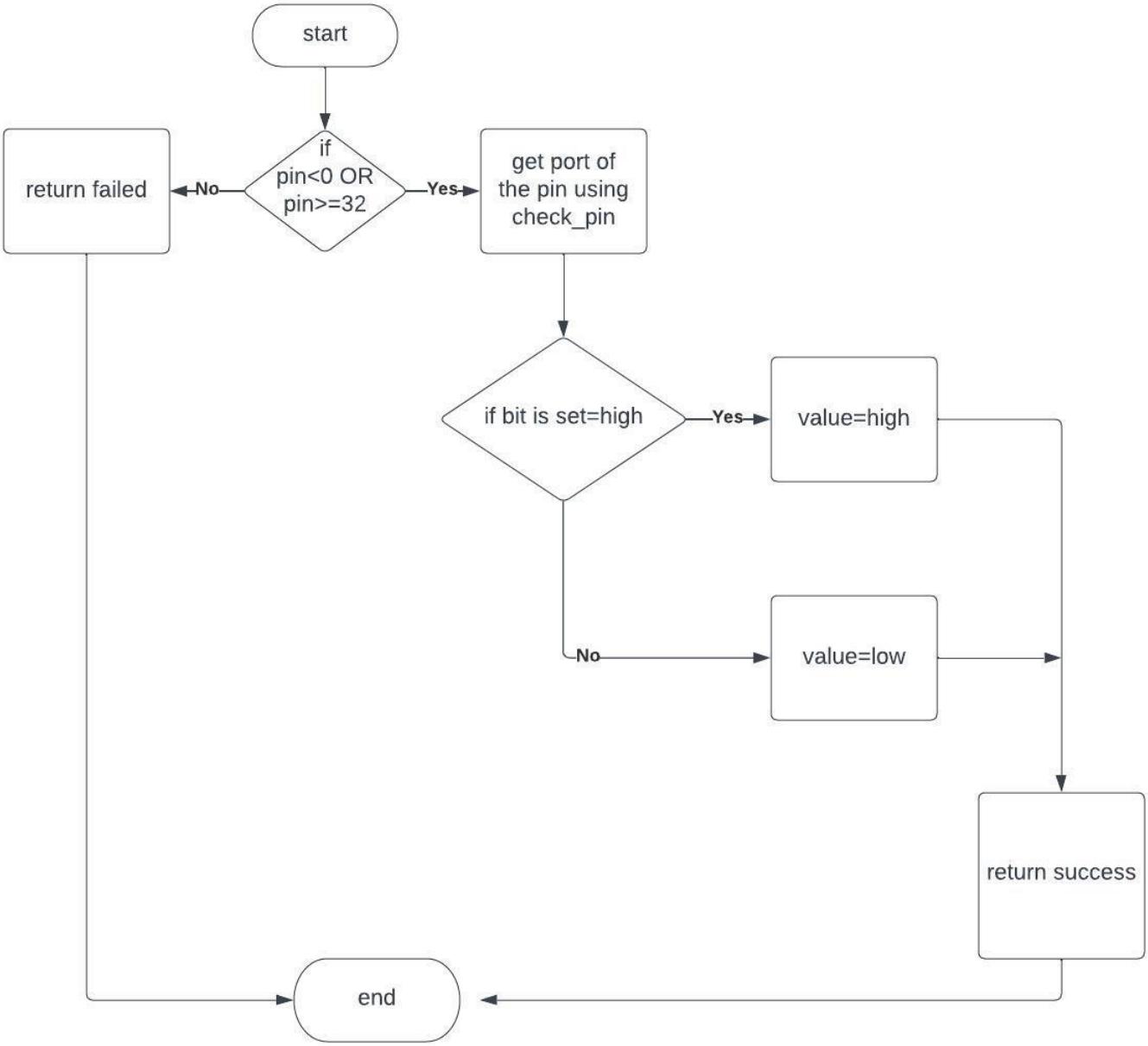
APIs flowcharts

- EN_STATE **digitalWrite(uint8 pin_no,EN_PIN_VALUE pin_val);**



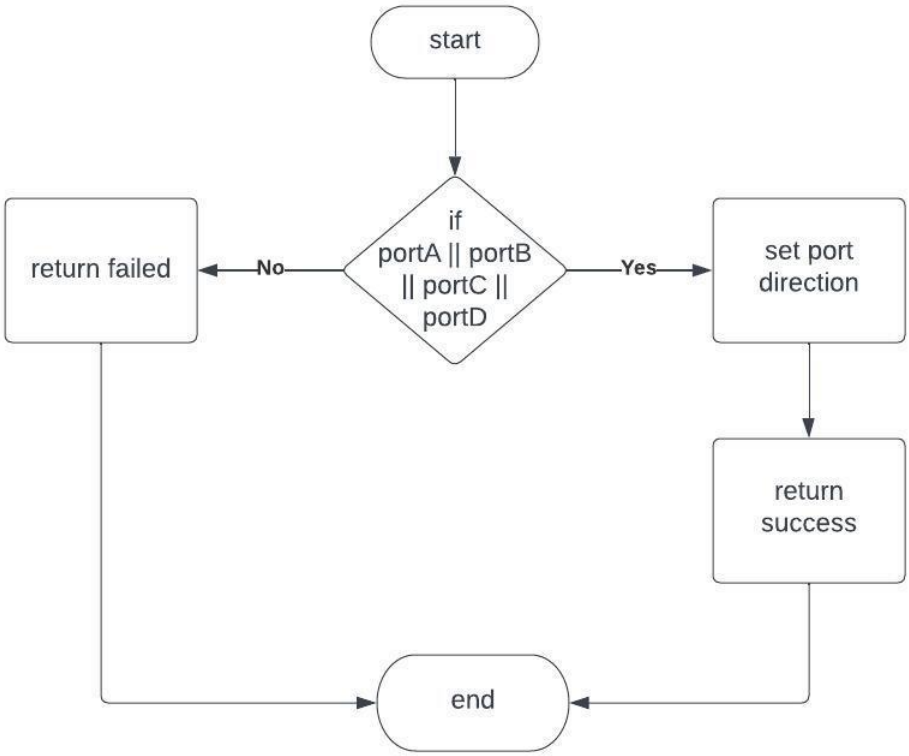
APIs flowcharts

- EN_STATE **digitalRead(uint8 pin_no,uint8 *pin_val);**

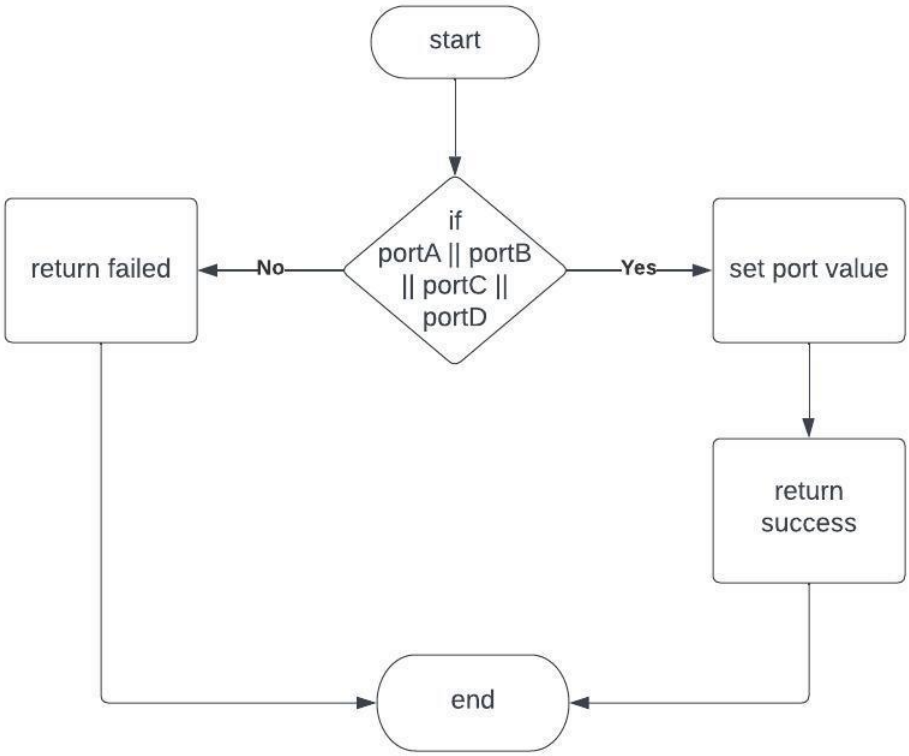


APIs flowcharts

EN_STATE portMode(REG_NAME port,EN_PORT_DIRECTION port_direction);

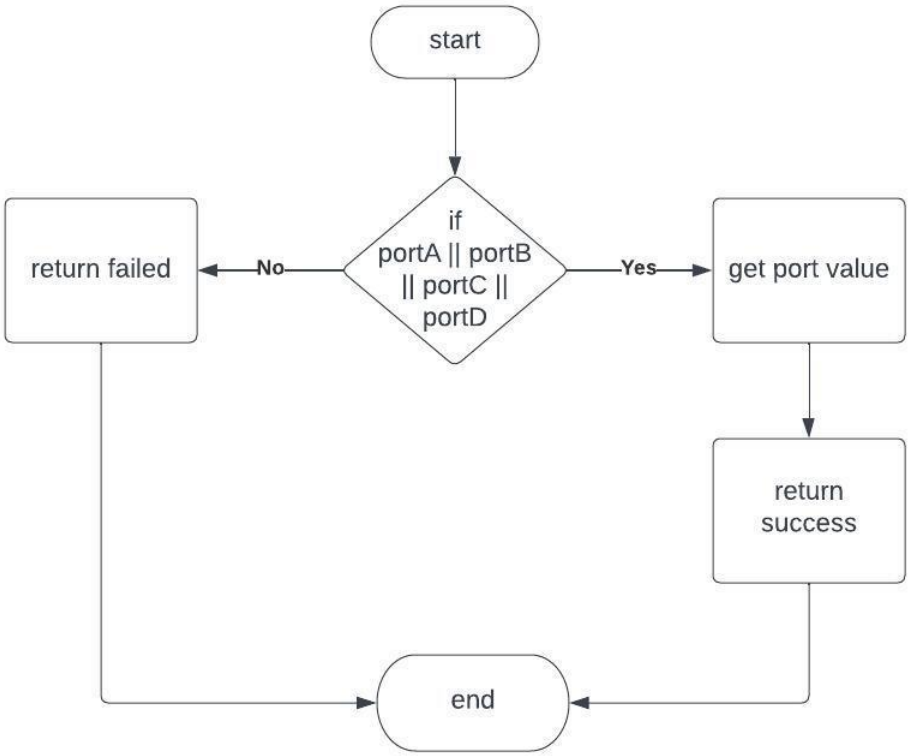


EN_STATE digitalWrite_Port(REG_NAME port,EN_PORT_VALUE port_val);

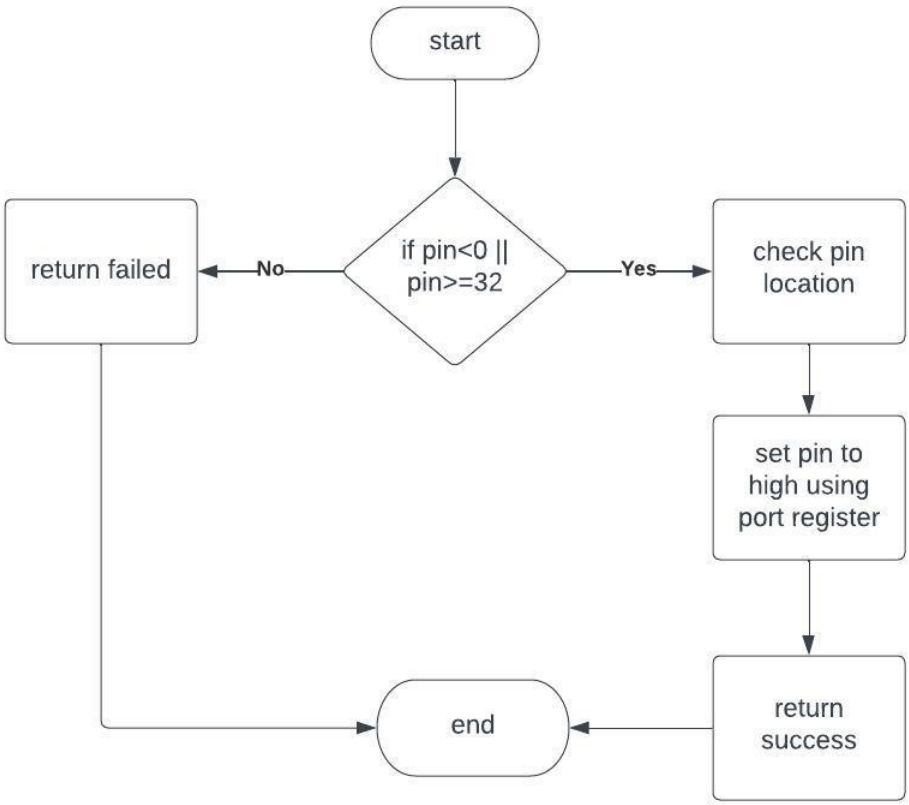


APIs flowcharts

EN_STATE digitalRead_Port(REG_NAME port,uint8 *port_val);

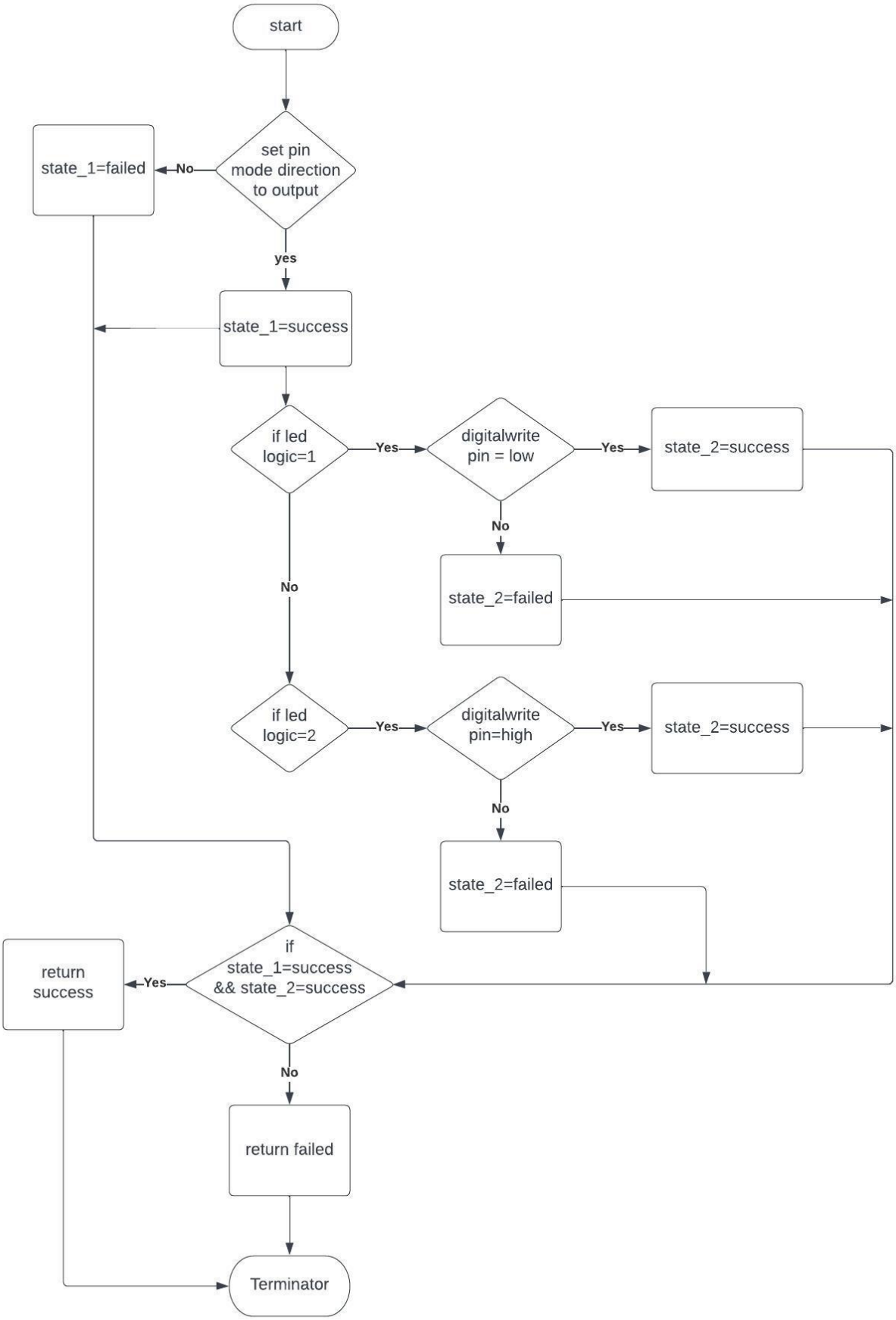


EN_STATE **Enable_PULLUP (uint8 pin_no);**



APIs flowcharts

EN_STATE **LED_init(uint8 led);**



APIs flowcharts

EN_STATE **LED_digitalwrite(uint8 led,EN_PIN_VALUE value);**

APIs flowcharts

EN_STATE Button_init(uint8 pin,EN_internal_pullup state);

uint8 **Button_Read(uint8 pin)**