# Moving Car Capstone Project


**By Team 2**

- Mohab Ahmed
- Anas Mahmoud
- Mustafa Mohammed
- Omar Taha

# Table of Contents

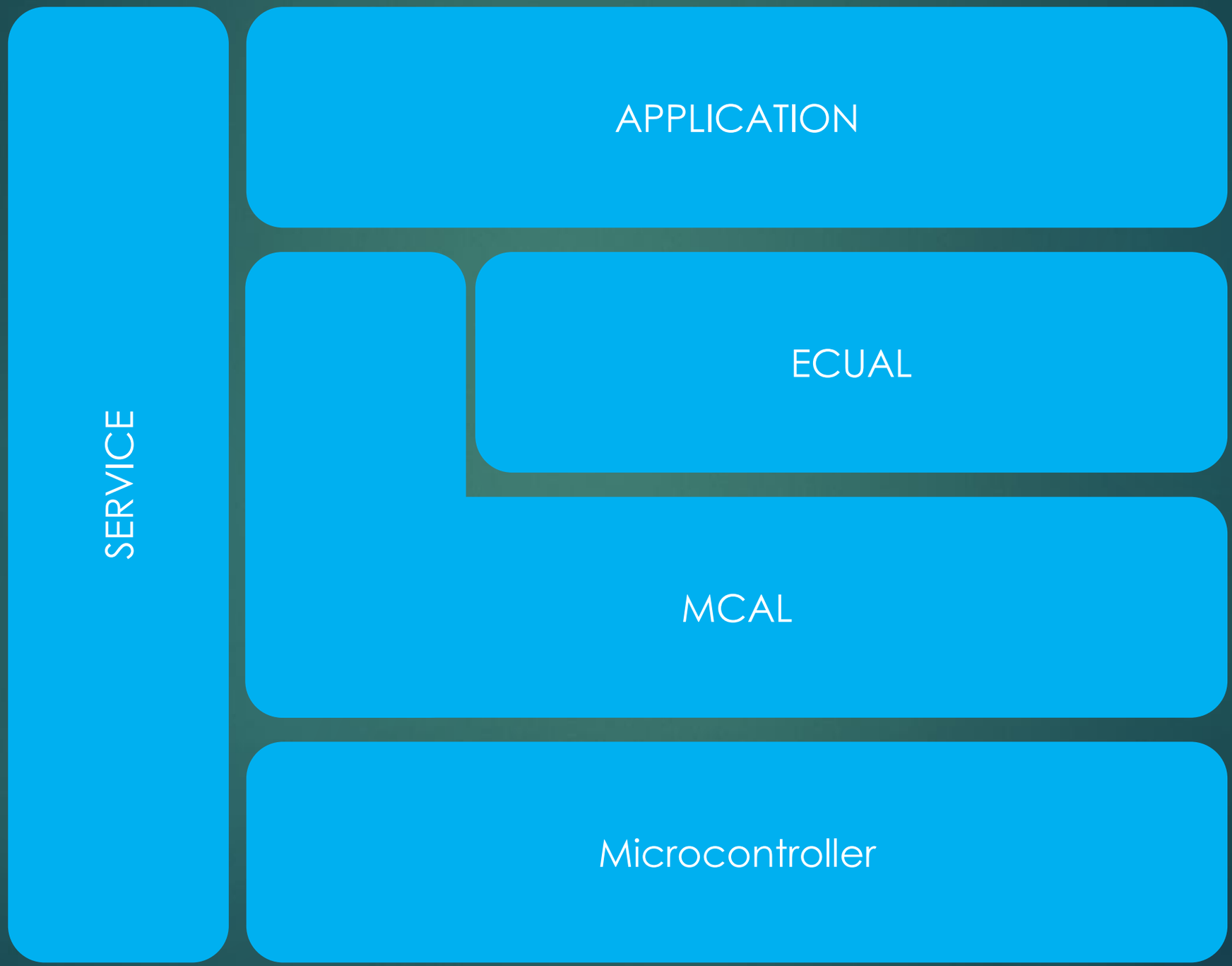
# Introduction

Description :

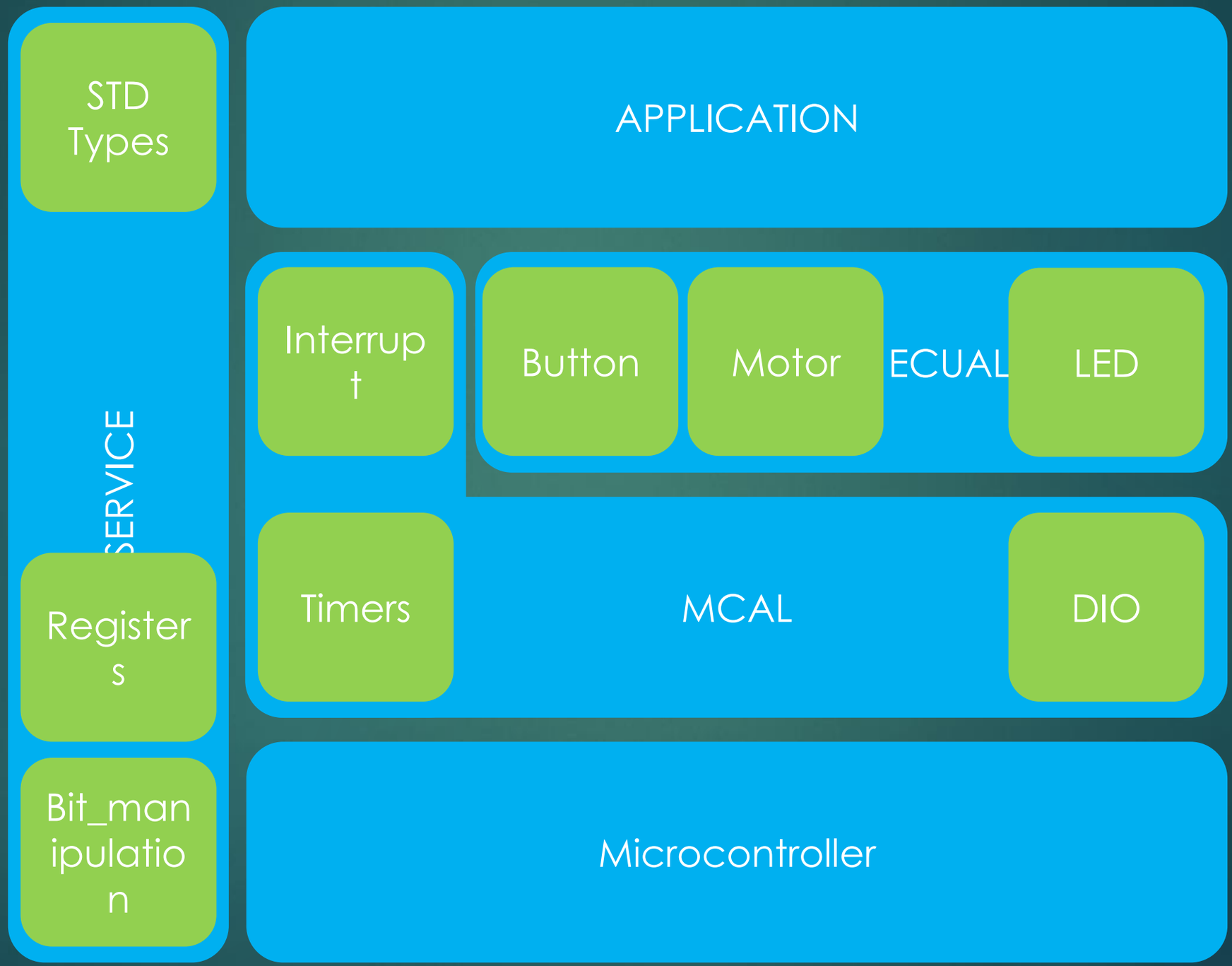
1. **Car Components**:
    1. Four motors (M1, M2, M3, M4)
    2. One button to start (PB1)
    3. One button for stop (PB2)
    4. Four LEDs (LED1, LED2, LED3, LED4)
2. **System Requirements**:
    1. The car starts initially from 0 speed
    2. When PB1 is pressed, the car will move forward after 1 second
    3. The car will move forward to create the longest side of the rectangle for 3 seconds with 50% of its maximum speed
    4. After finishing the first longest side the car will stop for 0.5 seconds, rotate 90 degrees to the right, and stop for 0.5 second
    5. The car will move to create the short side of the rectangle at 30% of its speed for 2 seconds
    6. After finishing the shortest side the car will stop for 0.5 seconds, rotate 90 degrees to the right, and stop for 0.5 second
    7. Steps 3 to 6 will be repeated infinitely until you press the stop button (PB2)
    8. PB2 acts as a sudden break, and it has the highest priority

# Project flowchart

# Layered Architecture:



# Modules/Drivers:



# APIs:

**Motor Driver:**

- err_state MOTOR_init(uint8_t u8_a_pinNumber, uint8_t u8_a_portNumber);
- err_state MOTOR_on(uint8_t u8_a_mask, uint8_t u8_a_portNumber);
- err_state MOTOR_off(uint8_t u8_a_mask, uint8_t u8_a_portNumber);
- err_state MOTOR_control(uint8_t u8_a_mask, uint8_t u8_a_portNumber,float f_a_speedPercentage);

**Button Driver:**

- err_state BUTTON_init(uint8_t u8_a_pinNumber, uint8_t u8_a_portNumber);
- err_state BUTTON_read(uint8_t u8_a_pinNumber, uint8_t u8_a_portNumber, pin_state *en_a_value);

**LED Driver:**

- err_state LED_init(uint8_t u8_a_ledPin, uint8_t u8_a_ledPort);
- err_state LED_on(uint8_t u8_a_ledPin, uint8_t u8_a_ledPort);
- err_state LED_off(uint8_t u8_a_ledPin, uint8_t u8_a_ledPort);
- err_state LED_toggle(uint8_t u8_a_ledPin, uint8_t u8_a_ledPort);
- err_state LED_blink(uint8_t u8_a_ledPin, uint8_t u8_a_ledPort,float f_a_onTime, float f_a_offTime);
- err_state LED_array_blink(uint8_t u8_a_mask, uint8_t u8_a_ledPort,float f_a_onTime, float f_a_offTime);
- err_state LED_array_on(uint8_t u8_a_mask, uint8_t u8_a_ledPort);
- err_state LED_array_off(uint8_t u8_a_mask, uint8_t u8_a_ledPort);

**DIO Driver:**

- err_state DIO_init(uint8_t u8_l_pinNumber, uint8_t u8_l_portNumber, pin_dir en_l_direction);
- err_state DIO_write(uint8_t u8_l_pinNumber, uint8_t u8_l_portNumber, pin_state en_l_value);
- err_state DIO_toggle(uint8_t u8_l_pinNumber, uint8_t u8_l_portNumber);
- err_state DIO_read(uint8_t u8_l_pinNumber, uint8_t u8_l_portNumber, pin_state *en_l_value);err_state DIO_array_write(uint8_t mask, uint8_t portNumber, pin_state value);
- err_state DIO_array_write(uint8_t u8_l_mask, uint8_t u8_l_portNumber, pin_state en_l_value);

# APIs:

**Timers Driver:**

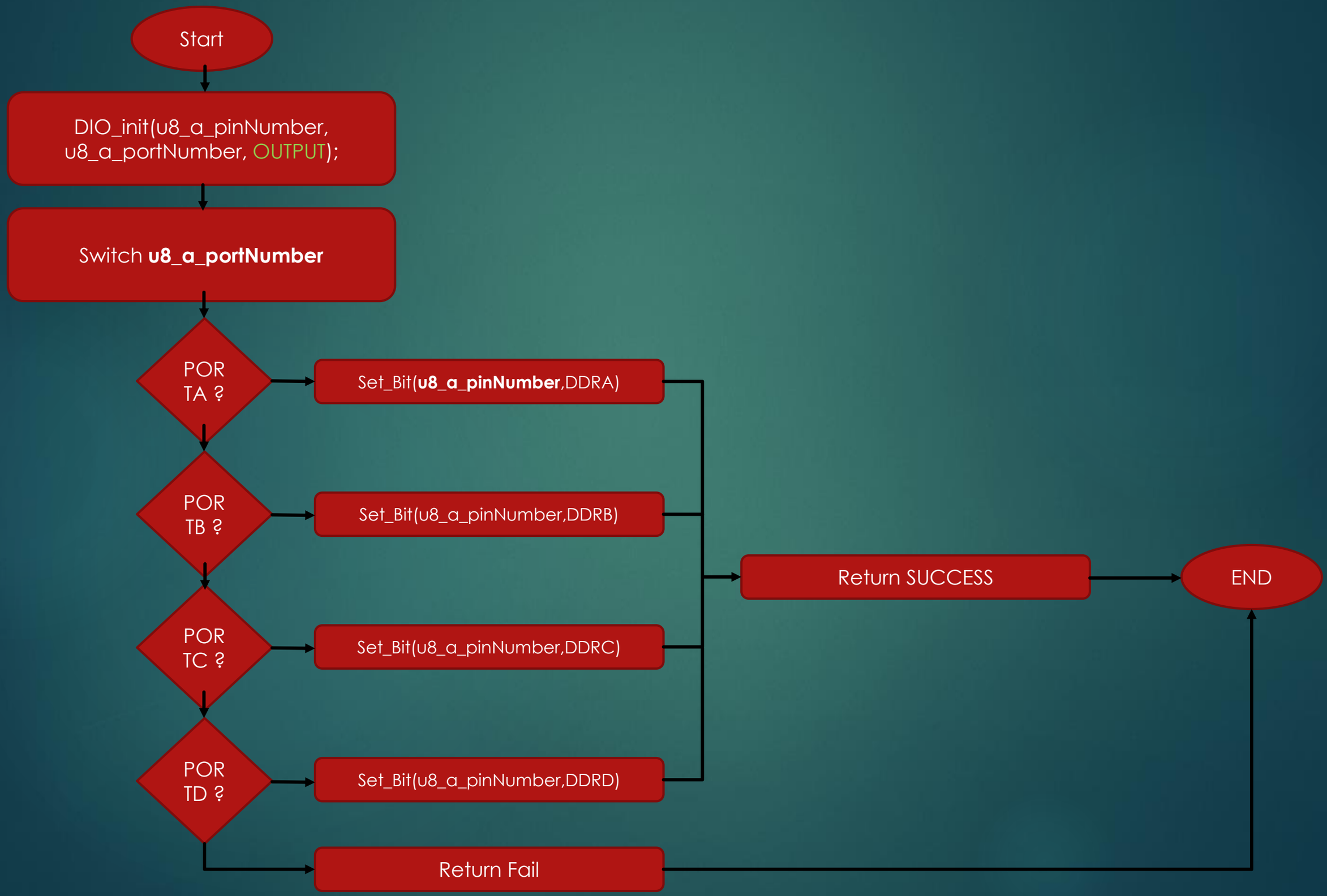
- err_state TIMER0_normalMode(void);
- err_state TIMER0_initialValue(uint8_t value);
- err_state TIMER0_prescalerMode(unsigned int u16_a_prescaler)
- err_state TIMER0_delay(float f_a_delayInMillis);
- unsigned int TIMER0_getInitialValue(float f_a_delayInMillis);
- err_state TIMER2_normalMode(void);
- err_state TIMER2_initialValue(uint8_t value);
- err_state TIMER2_perscalerMode(unsigned int prescaler);
- err_state TIMER2_delay(float f_a_delayInMillis);
- unsigned int TIMER2_getInitialValue(float f_a_delayInMillis);

**Interrupts Driver:**

- err_state BUTTON_init(uint8_t pinNumber, uint8_t portNumber);
- err_state BUTTON_read(uint8_t pinNumber, uint8_t portNumber, uint8_t *value);

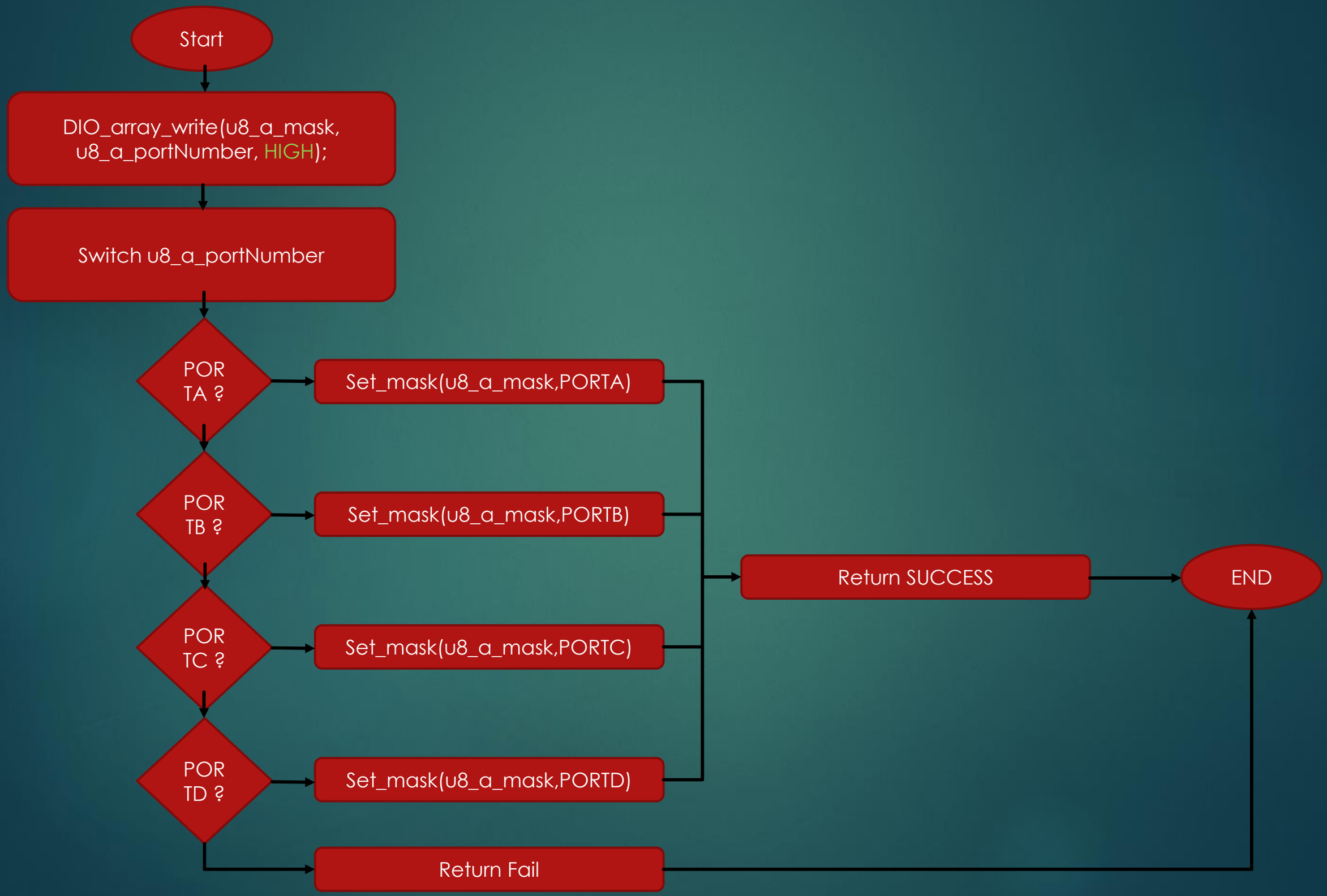
# APIs flowchart:

**err_state MOTOR_init(uint8_t u8_a_pinNumber, uint8_t u8_a_portNumber);**



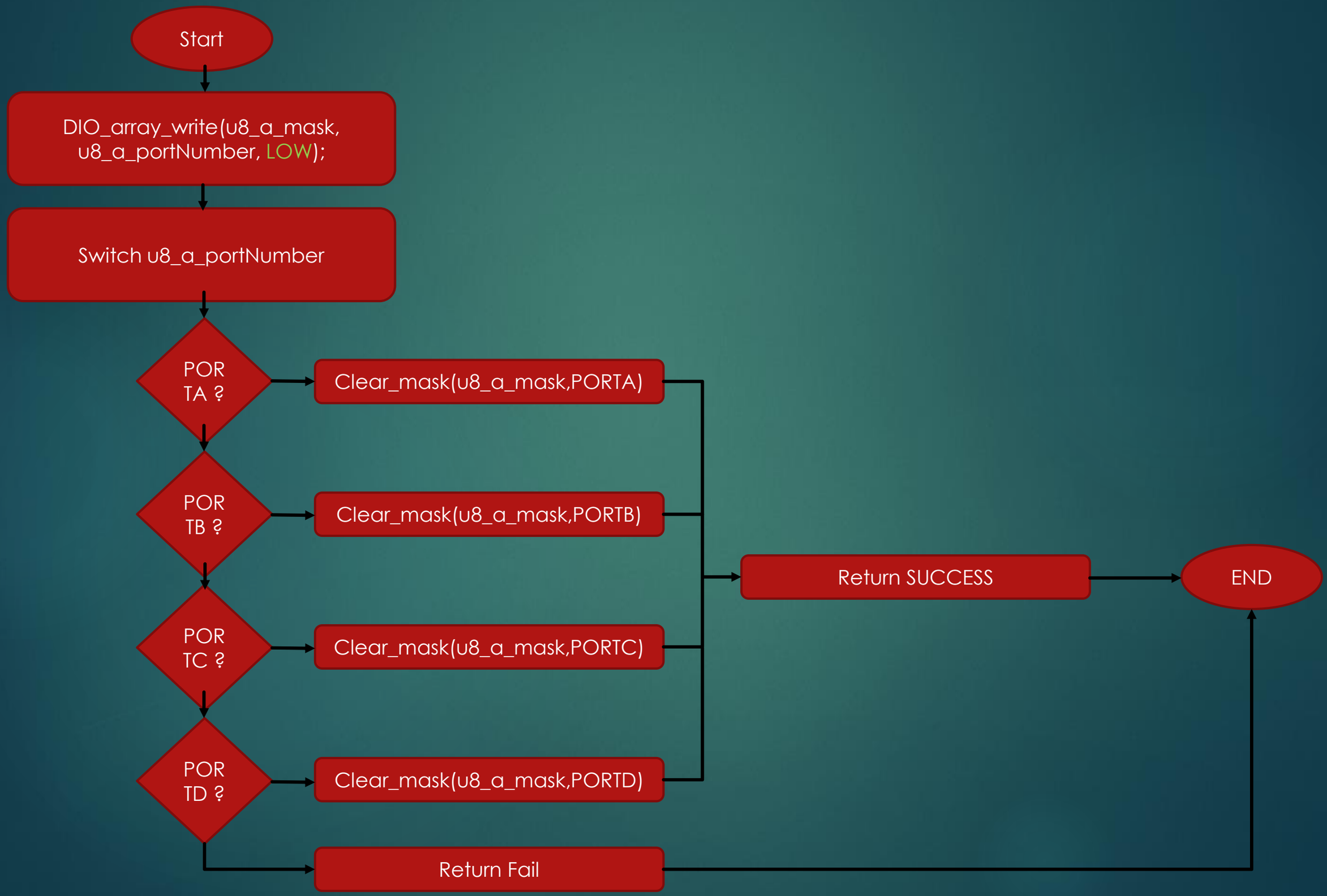
# APIs flowchart:

**err_state MOTOR_on(uint8_t u8_a_mask, uint8_t u8_a_portNumber);**



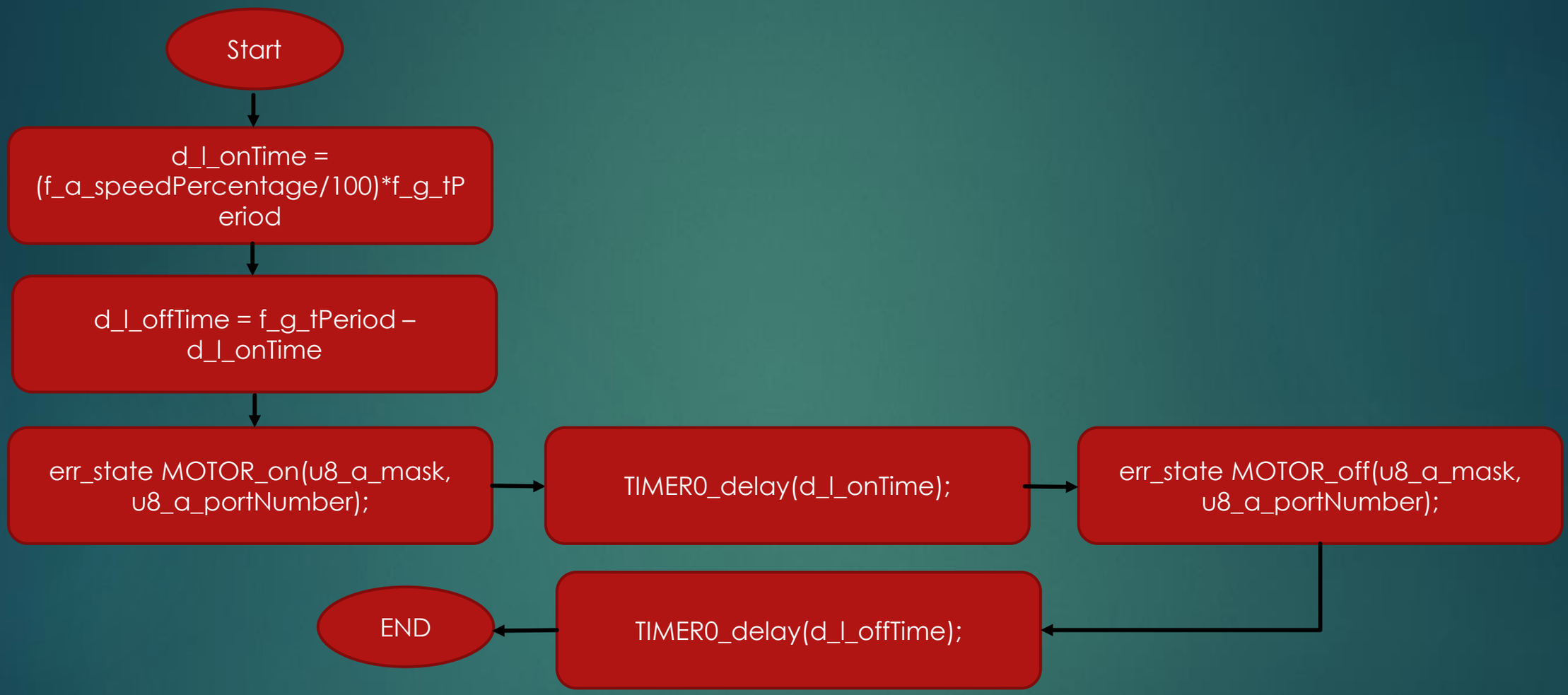
# APIs flowchart:

**err_state MOTOR_off(uint8_t u8_a_mask, uint8_t u8_a_portNumber);**



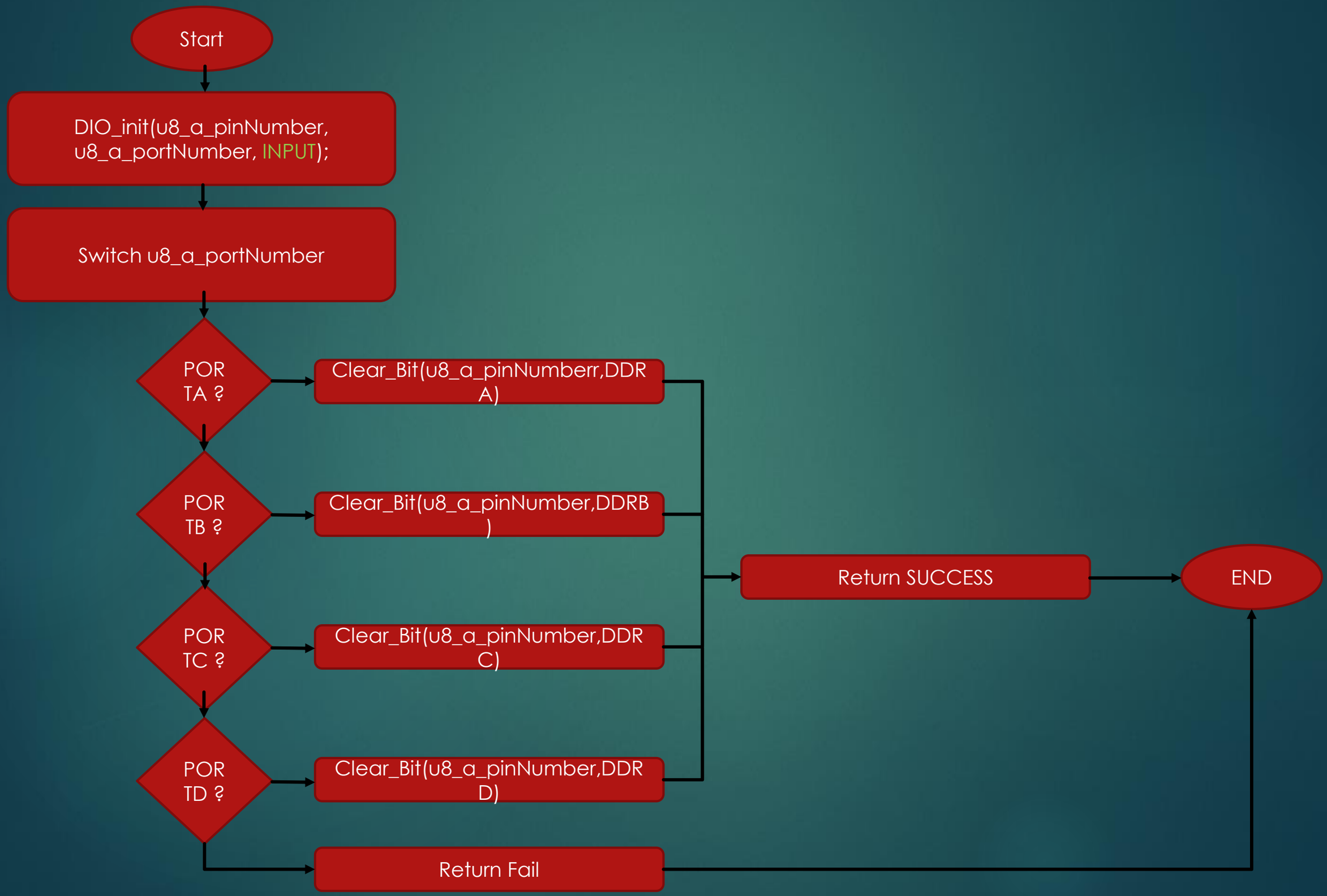
# APIs flowchart:

**err_state MOTOR_control(uint8_t u8_a_mask, uint8_t u8_a_portNumber,float f_a_speedPercentage);**



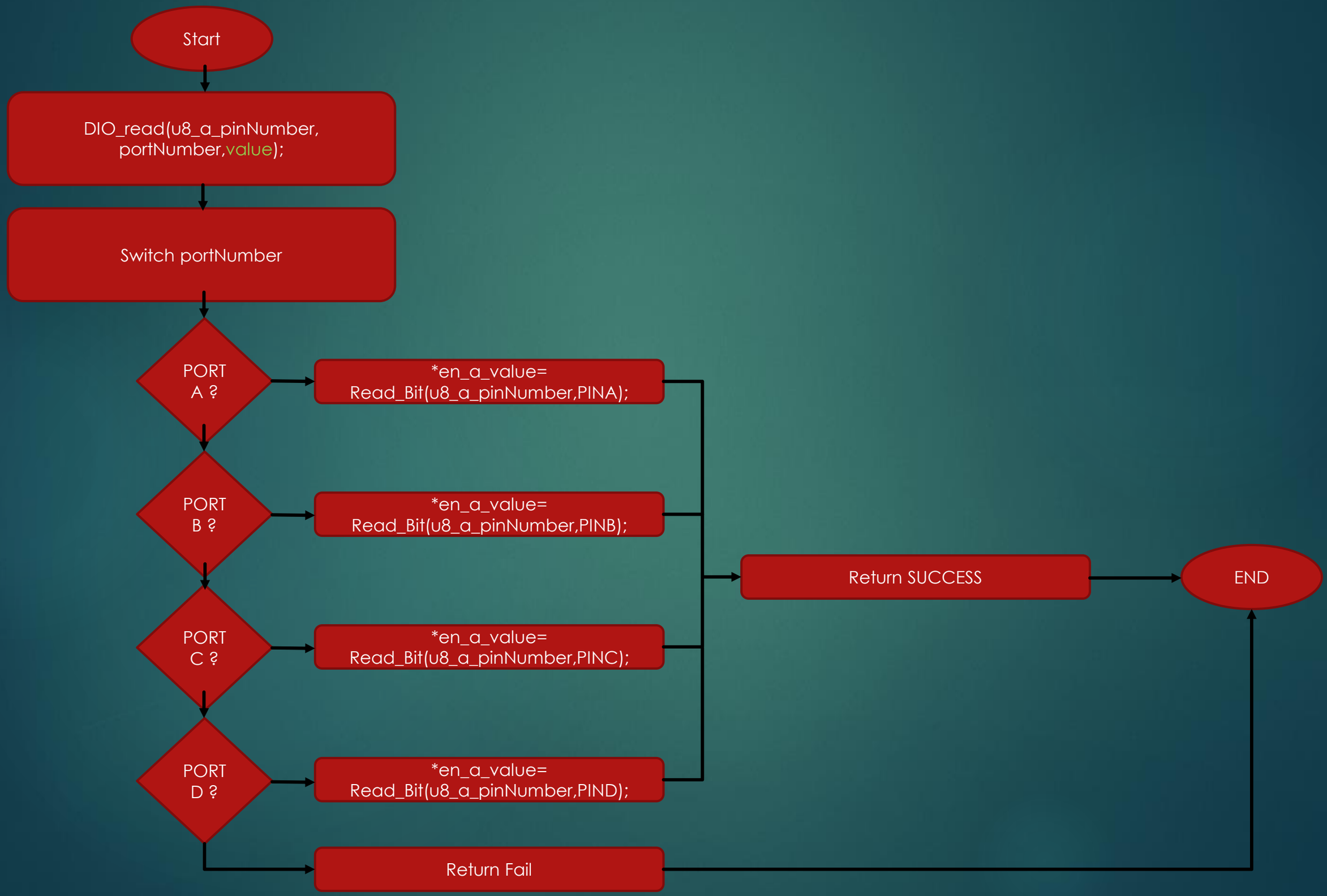
# APIs flowchart:

**err_state BUTTON_init(uint8_t u8_a_pinNumber, uint8_t u8_a_portNumber);**



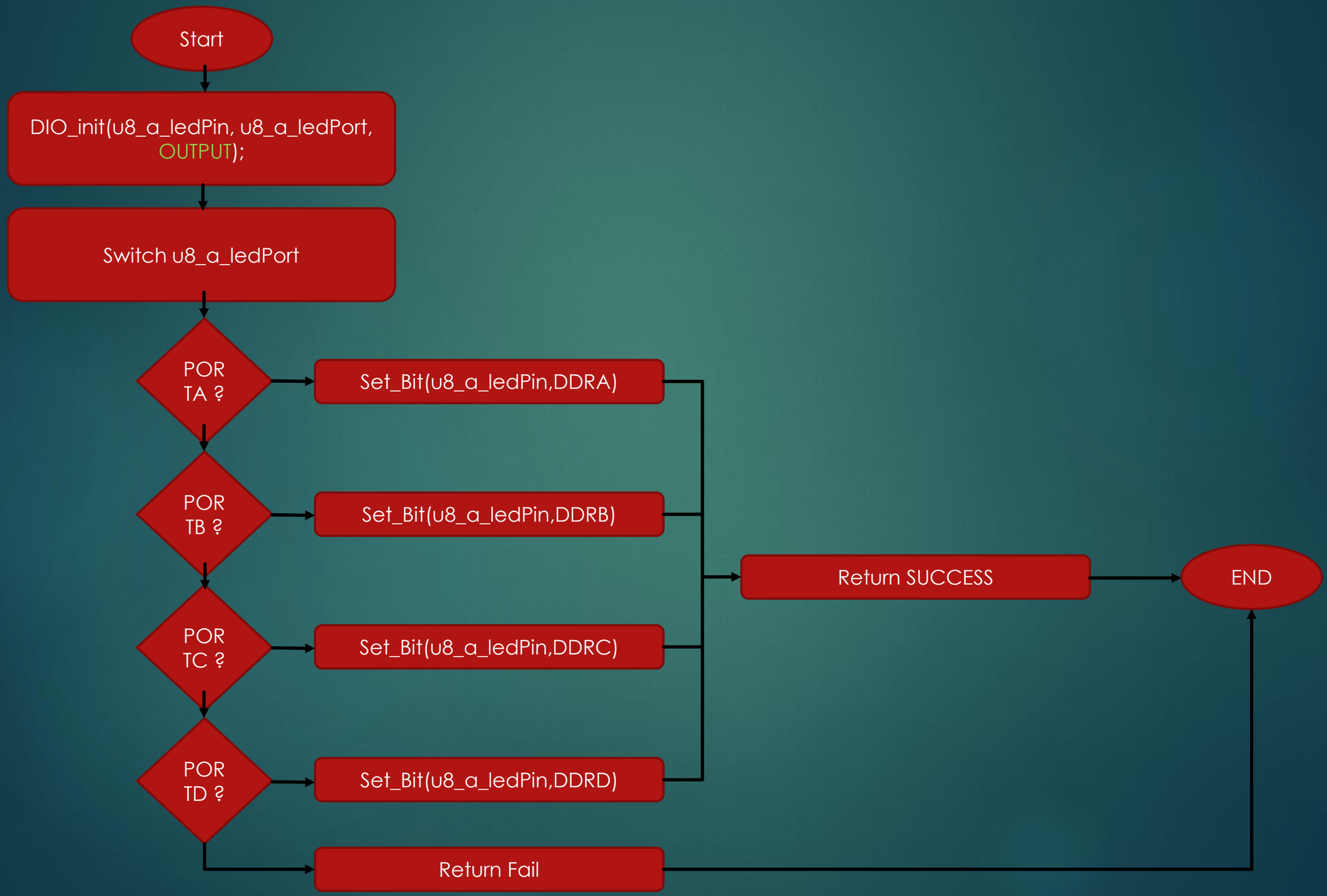
# APIs flowchart:

**err_state BUTTON_read(uint8_t u8_a_pinNumber, uint8_t u8_a_portNumber, pin_state *en_a_value);**



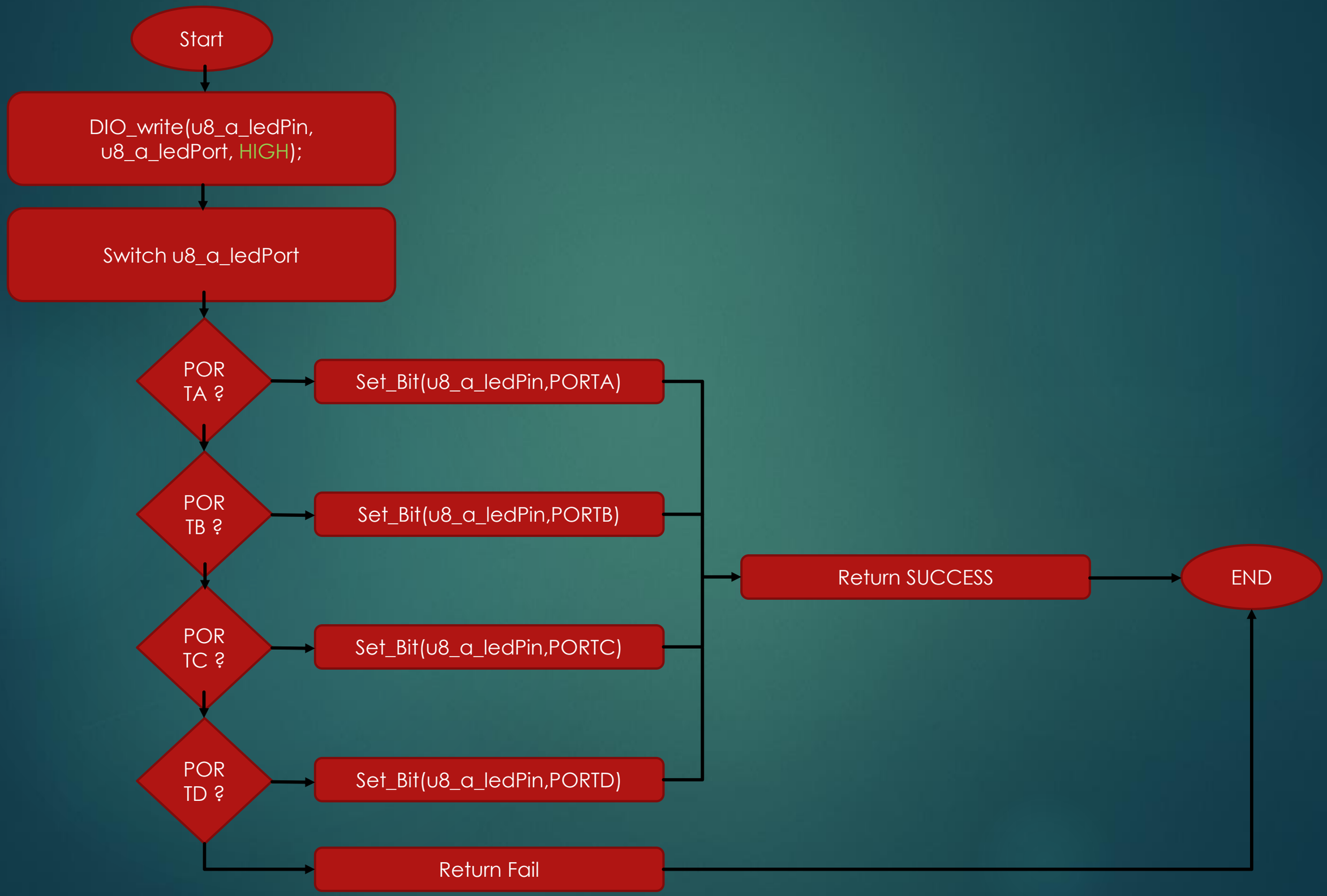
# APIs flowchart:

**err_state LED_init(uint8_t u8_a_ledPin, uint8_t u8_a_ledPort);**



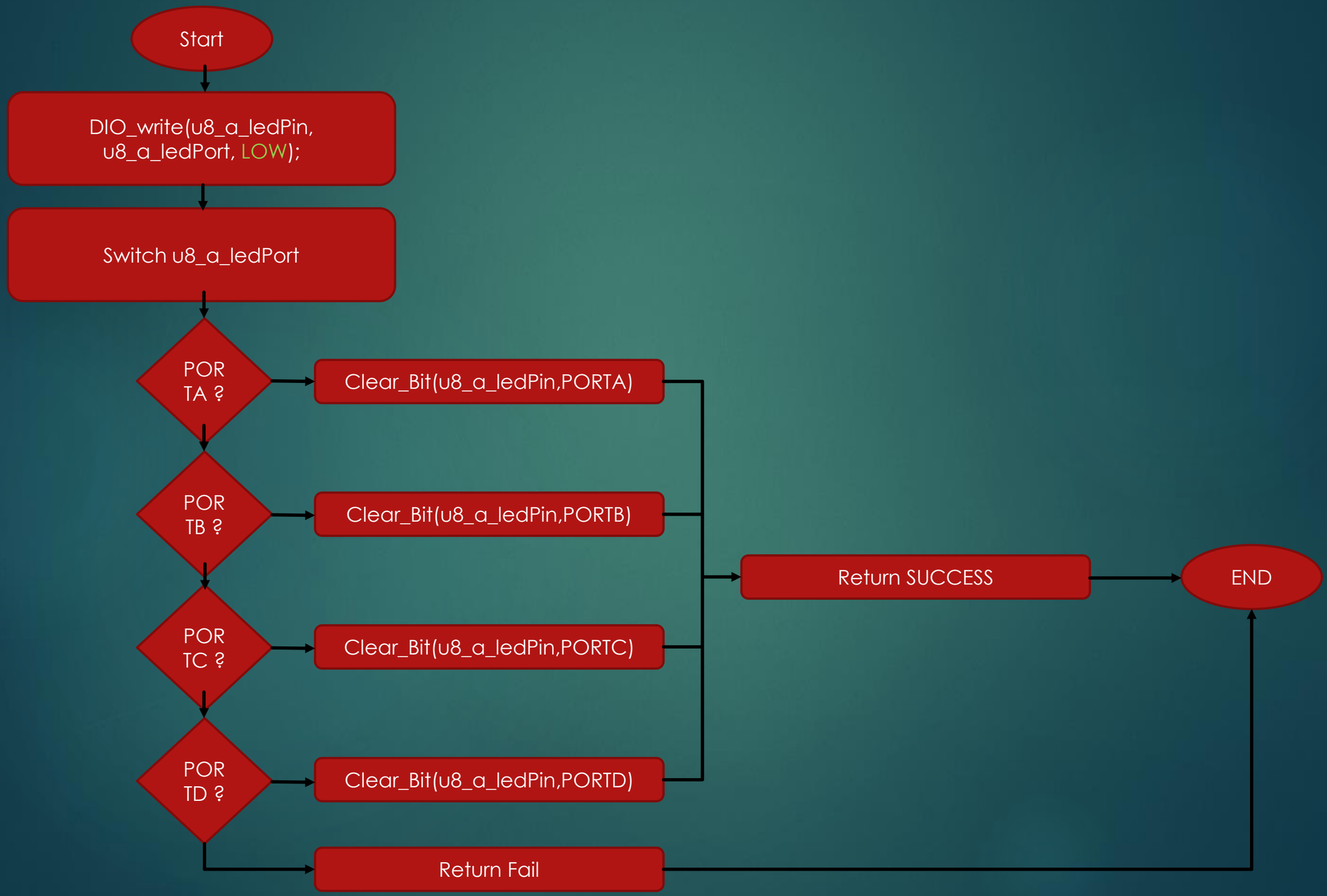
# APIs flowchart:

**err_state LED_on(uint8_t u8_a_ledPin, uint8_t u8_a_ledPort);**



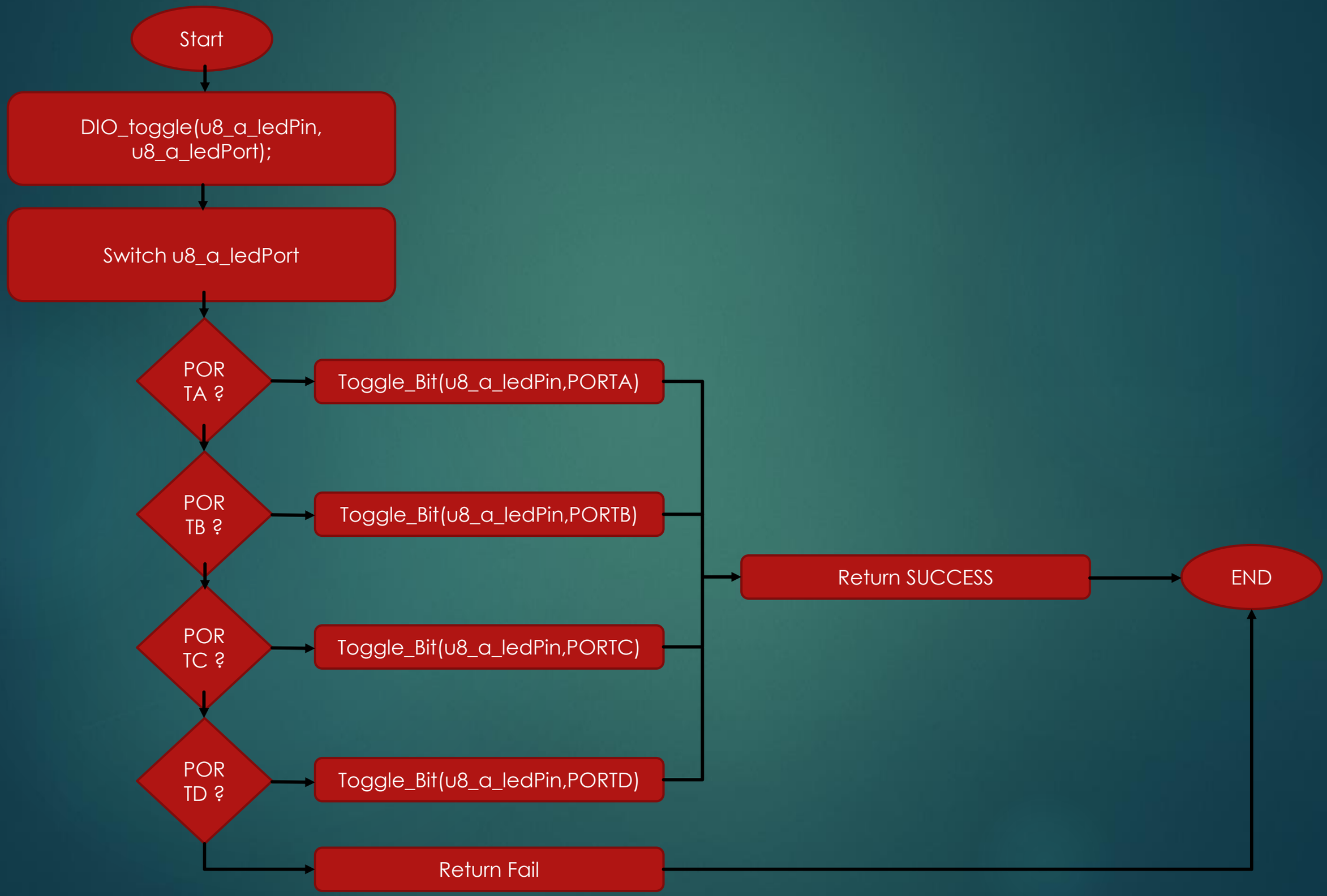
# APIs flowchart:

**err_state LED_off(uint8_t u8_a_ledPin, uint8_t u8_a_ledPort);**



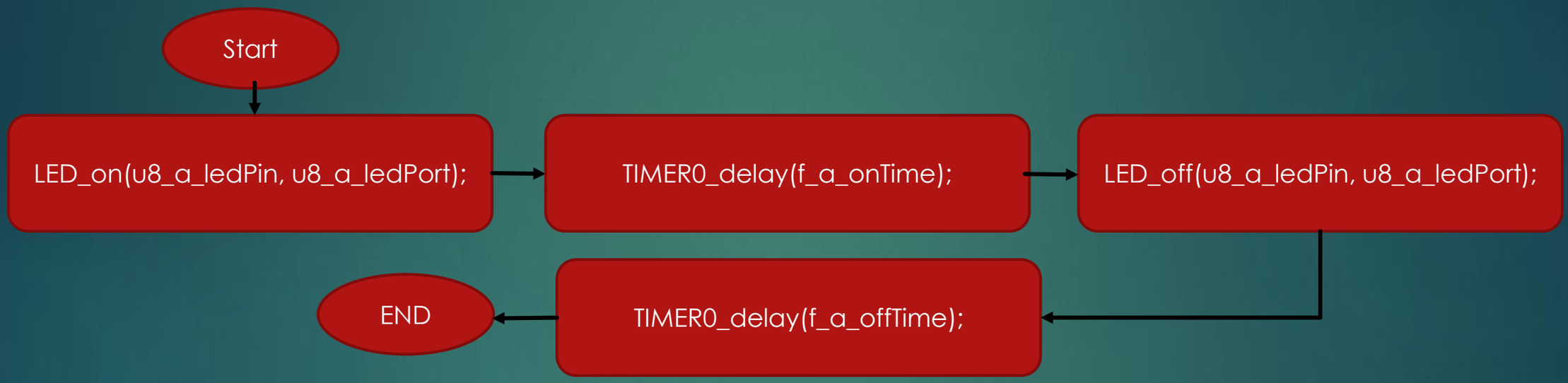
# APIs flowchart:

**err_state LED_toggle(uint8_t u8_a_ledPin, uint8_t u8_a_ledPort);**



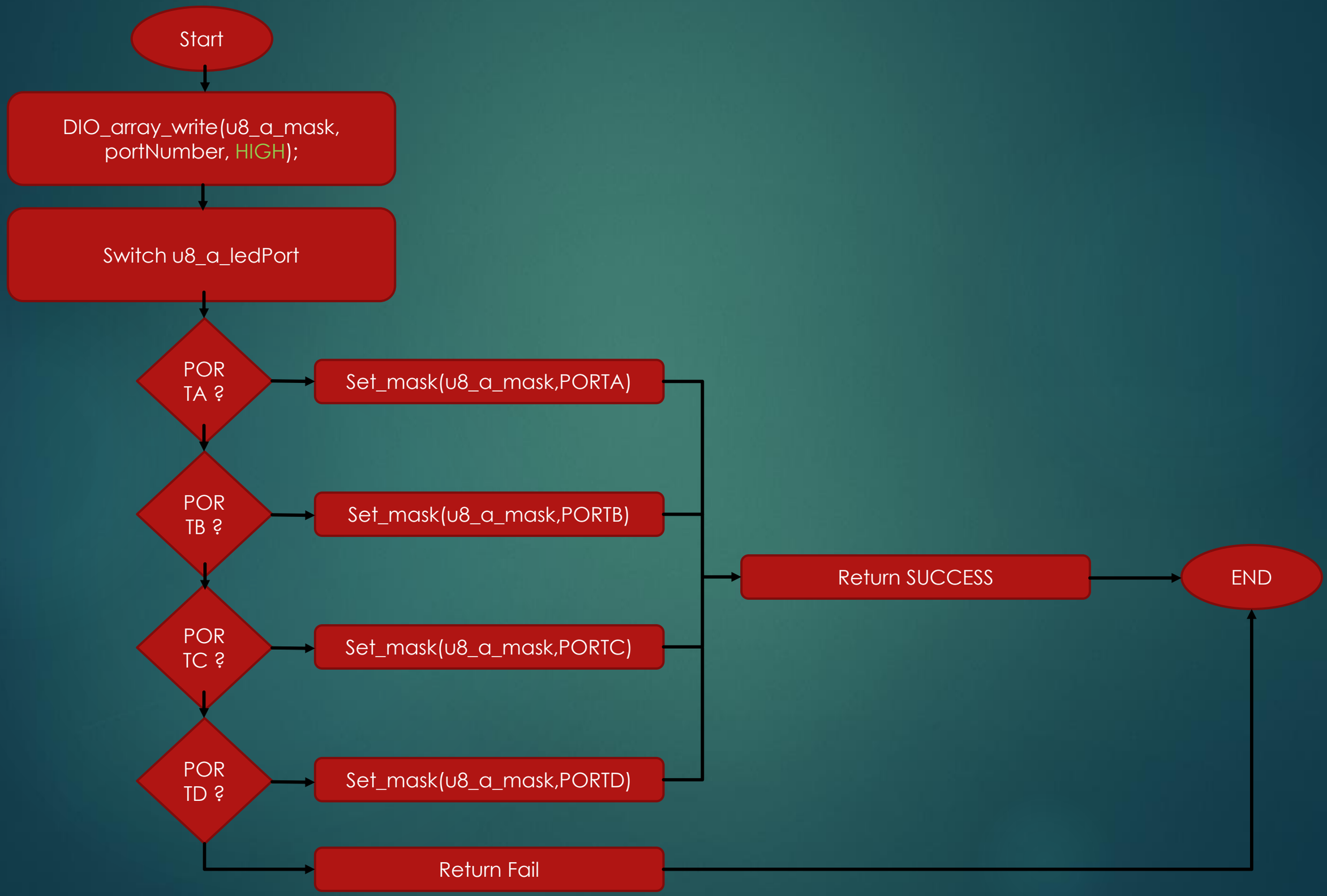
# APIs flowchart:

**err_state LED_blink(uint8_t u8_a_ledPin, uint8_t u8_a_ledPort,float f_a_onTime, float f_a_offTime);**



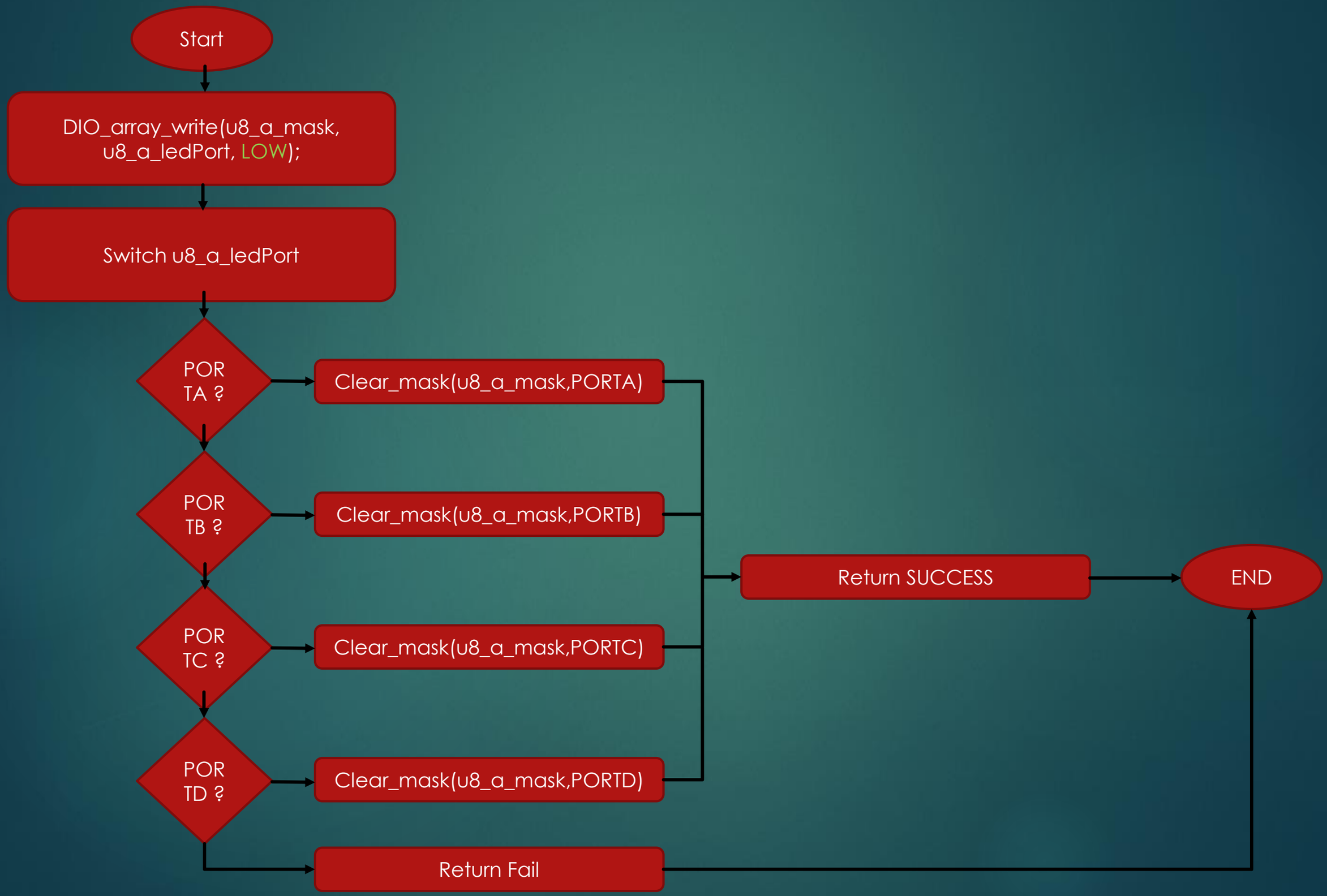
# APIs flowchart:

**err_state LED_array_on(uint8_t u8_a_mask, uint8_t u8_a_ledPort);**



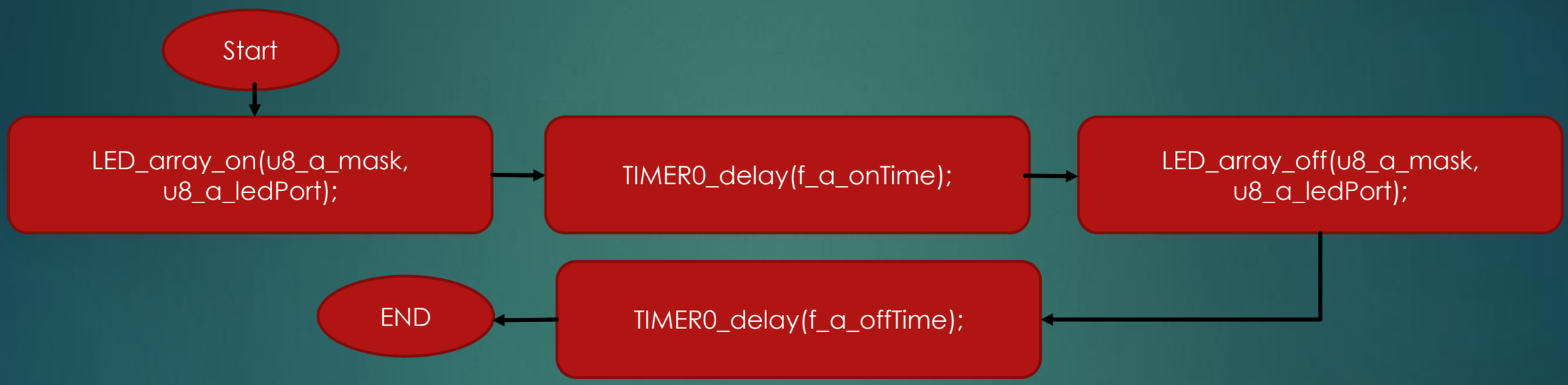
# APIs flowchart:

**err_state LED_array_off(uint8_t u8_a_mask, uint8_t u8_a_ledPort);**



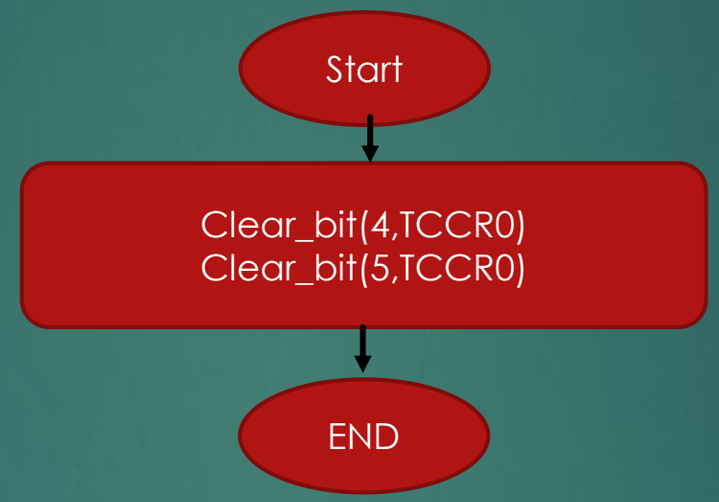
# APIs flowchart:

**err_state LED_array_blink(uint8_t u8_a_mask, uint8_t u8_a_ledPort,float f_a_onTime, float f_a_offTime);**



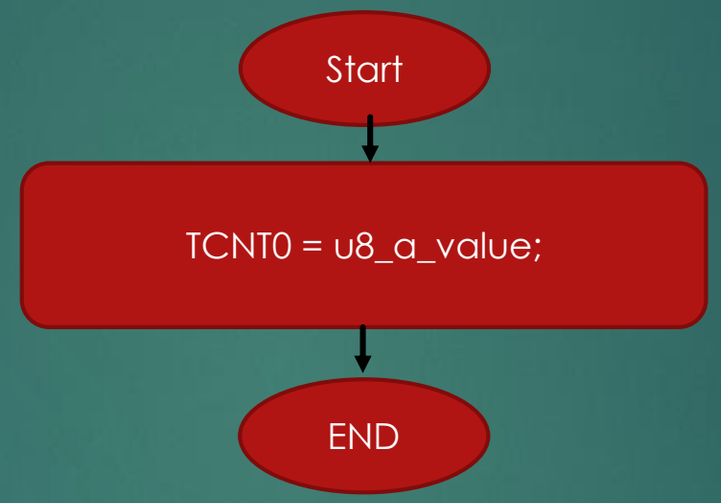
# APIs flowchart:

**err_state TIMER0_normalMode(void);**



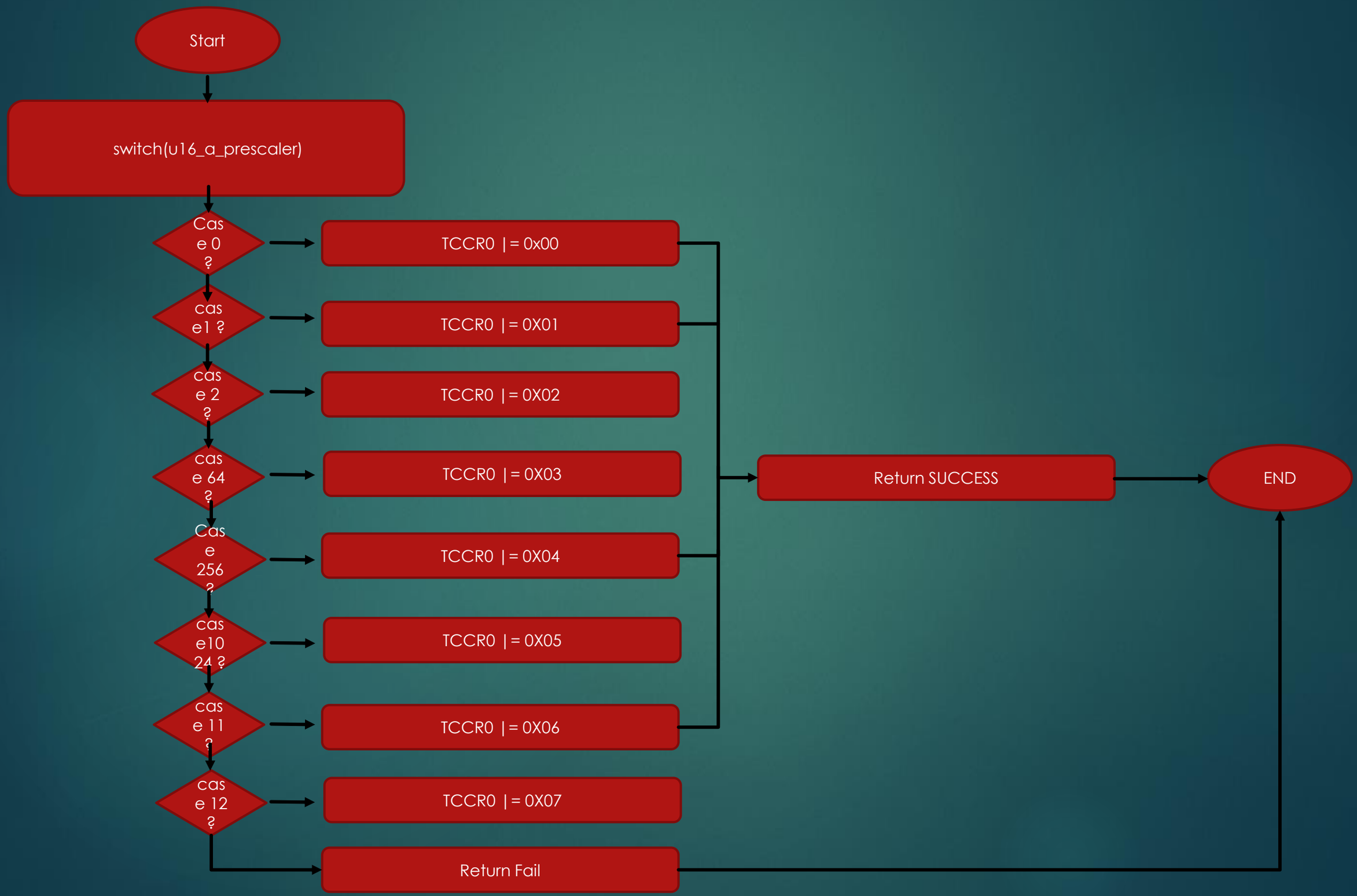
# APIs flowchart:

**err_state TIMER0_initialValue(uint8_t value);**



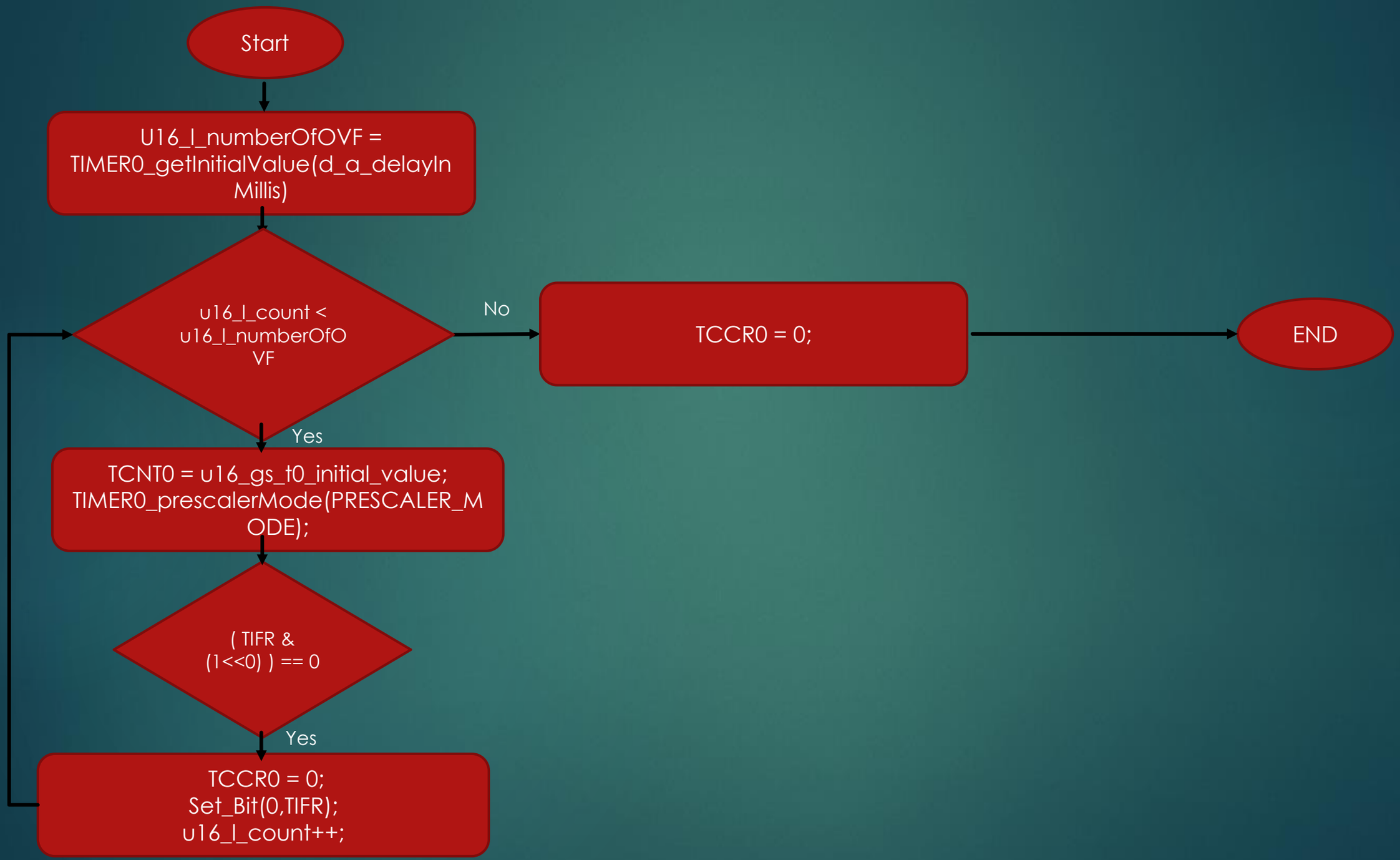
# APIs flowchart:

**err_state TIMER0_prescalerMode(unsigned int u16_a_prescaler)**



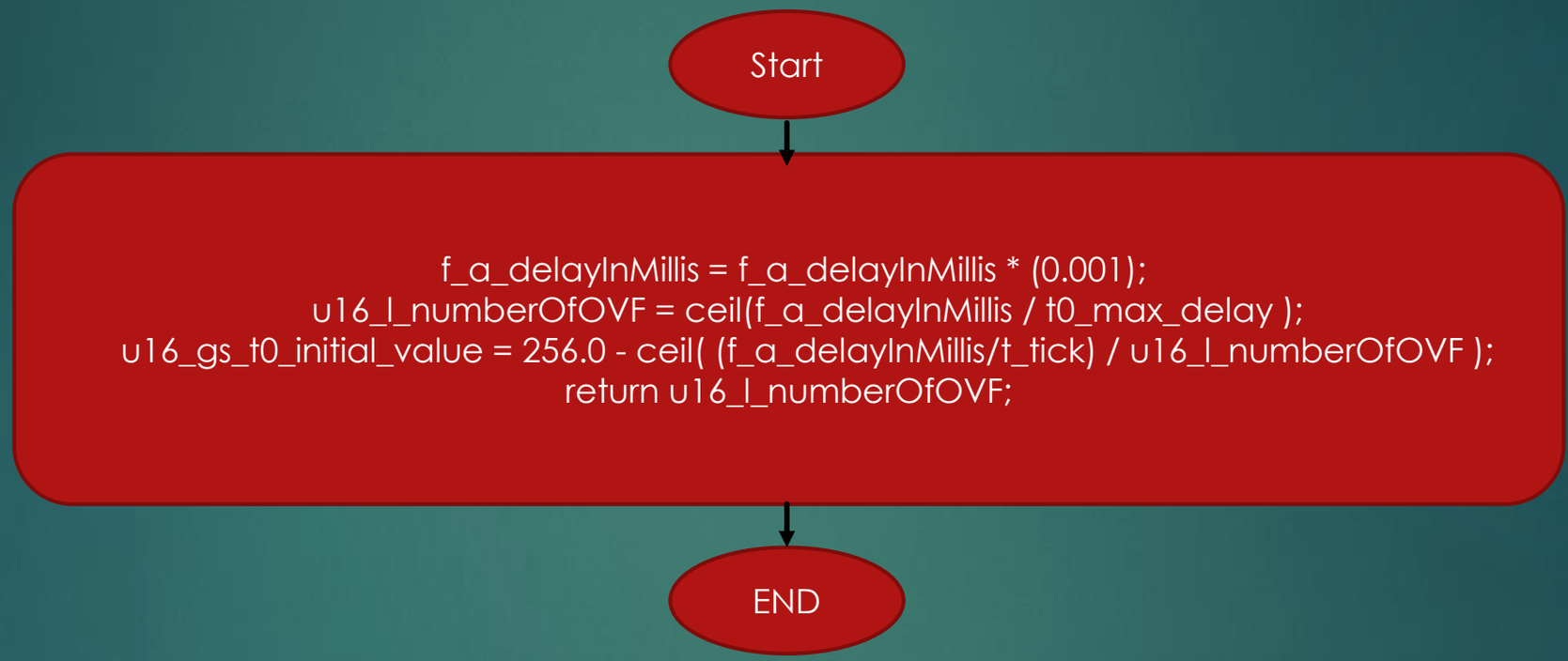
# APIs flowchart:

**err_state TIMER0_delay(float f_a_delayInMillis);**



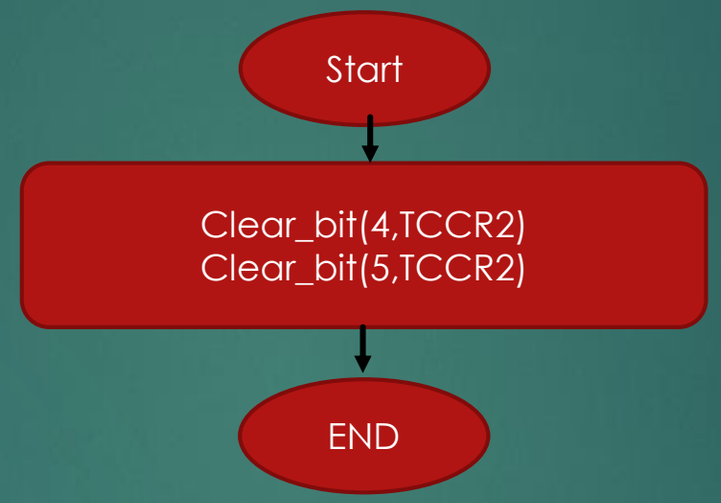
# APIs flowchart:

**unsigned int TIMER0_getInitialValue(float f_a_delayInMillis)**



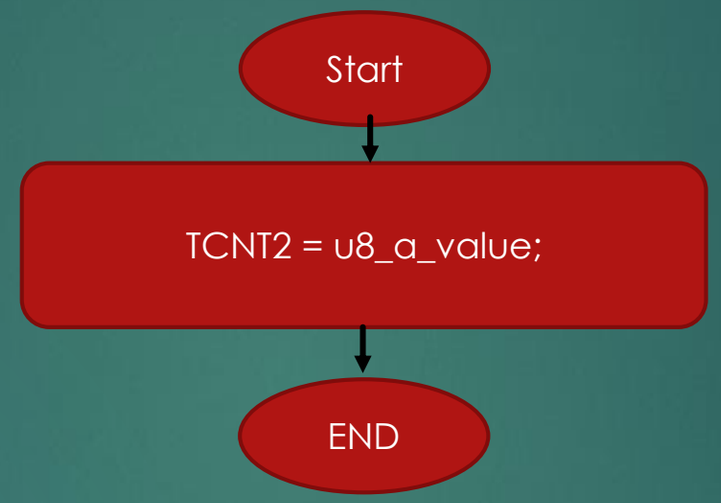
# APIs flowchart:

**err_state TIMER2_normalMode(void);**



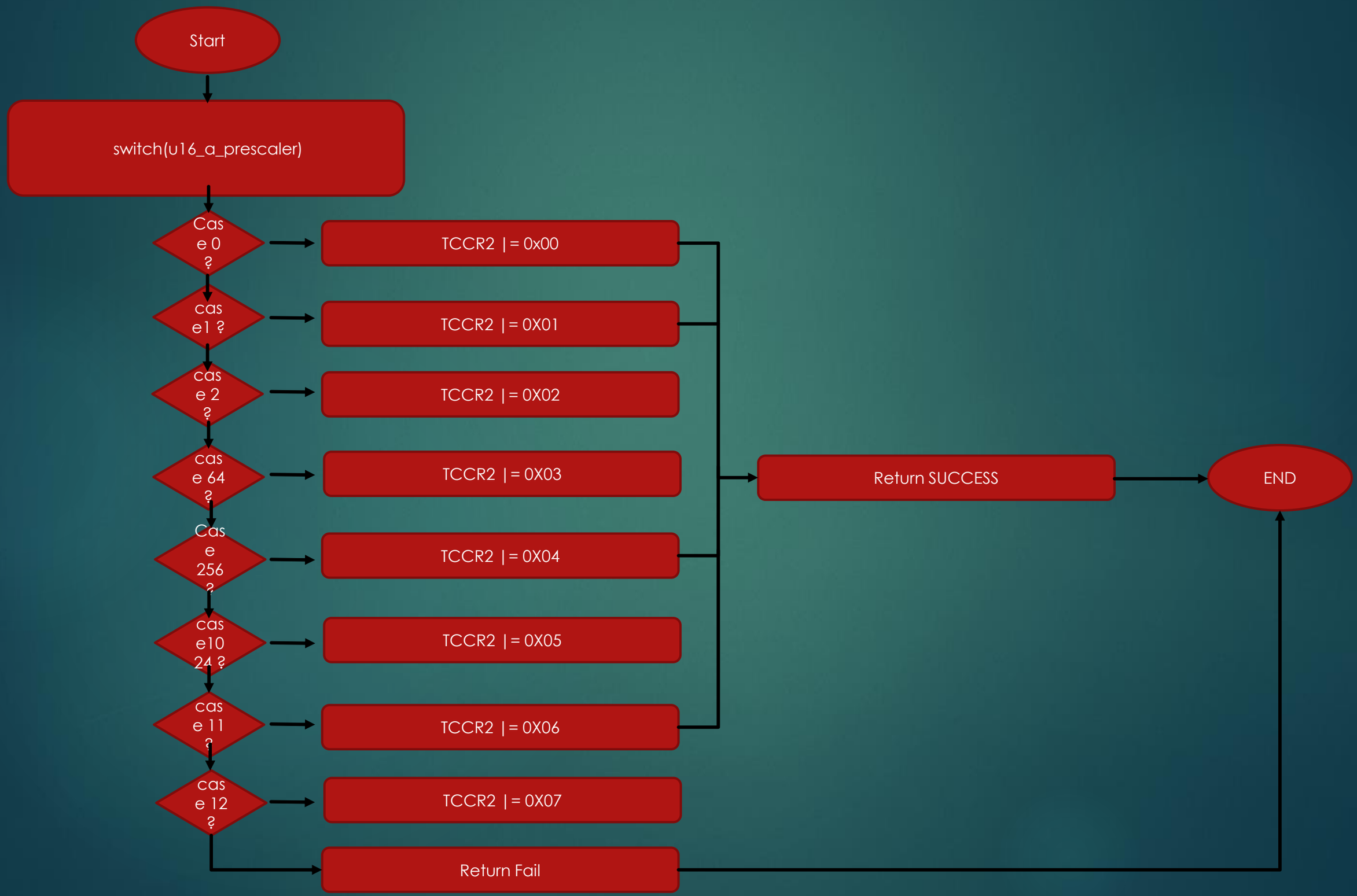
# APIs flowchart:

**err_state TIMER2_initialValue(uint8_t value);**



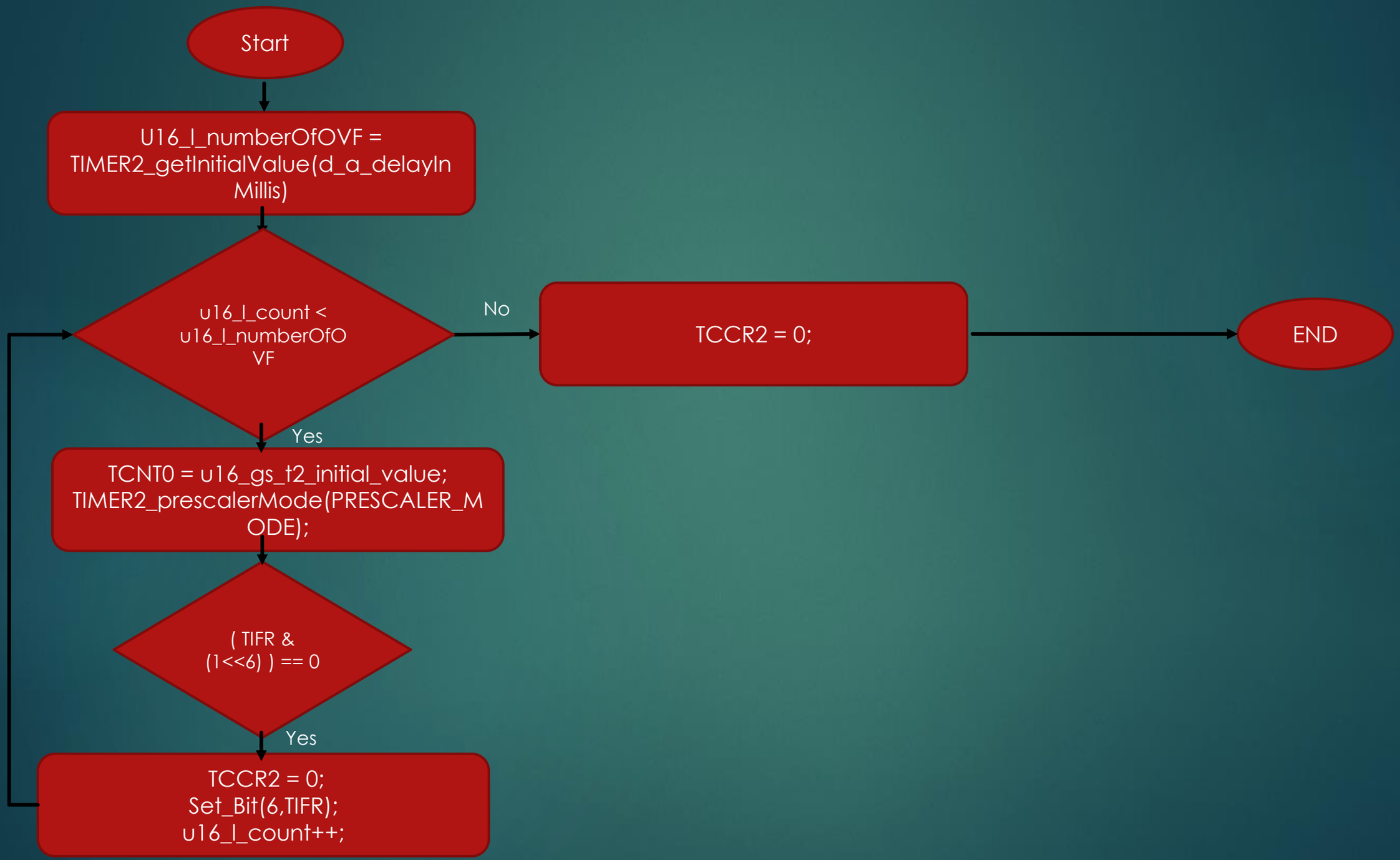
# APIs flowchart:

**err_state TIMER2_prescalerMode(unsigned int u16_a_prescaler)**



# APIs flowchart:

**err_state TIMER2_delay(float f_a_delayInMillis);**



# APIs flowchart:

**unsigned int TIMER2_getInitialValue(float f_a_delayInMillis)**