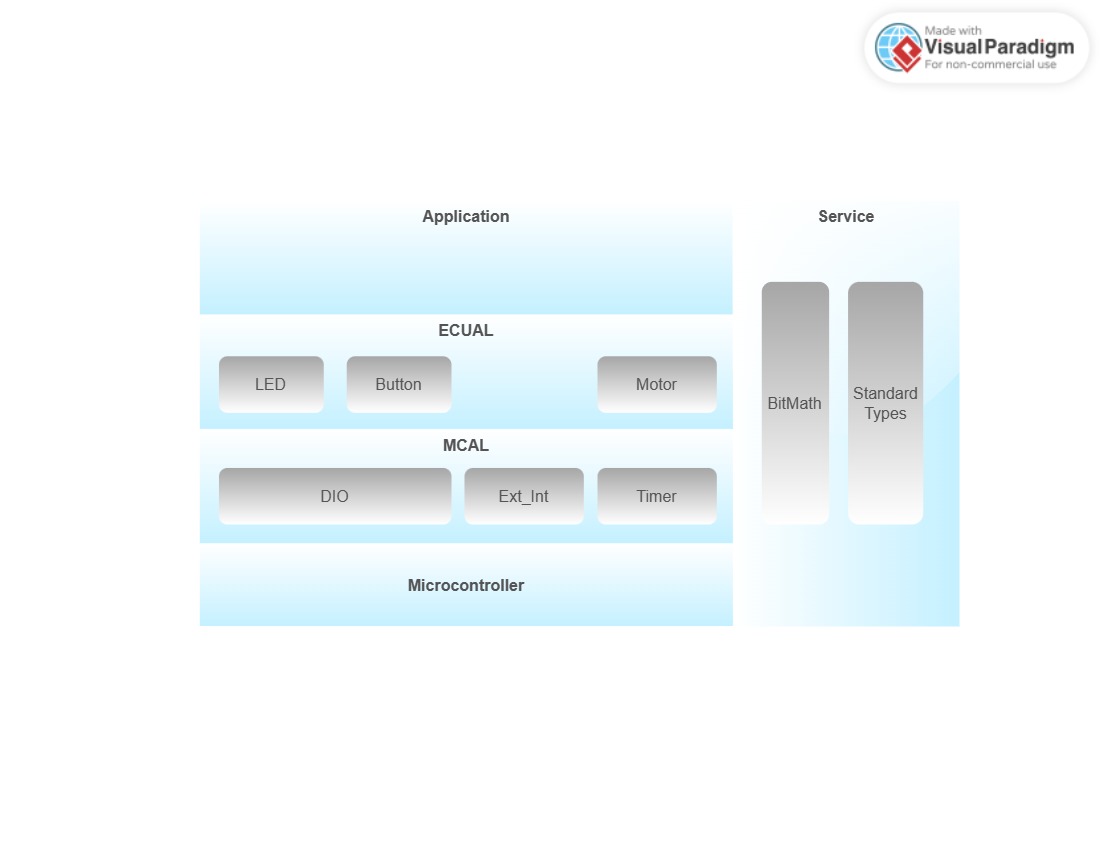
LED SEQUENCE v 3.0

# LAYERED ARCHITECTURE

MODULE AND PERIPHERALS DESCRIPTION

**DIO (Digital Input/Output):** This module deals with the digital input and output operations, such as reading and writing to digital pins of a microcontroller or a microprocessor. It may include functions for setting pin direction, reading and writing digital values, and handling interrupts related to digital pins.

**Motor:** This module is responsible for controlling the motors in the car, such as M1, M2, M3, and M4 as mentioned in the system requirements. It may include functions for setting motor speed, direction, and handling motor control signals.

**Button:** This module deals with the buttons in the system, such as PB1 and PB2 as mentioned in the system requirements. It may include functions for detecting button presses and handling button-related events.

**EXT\_INT (External Interrupt):** This module handles external interrupts, which are signals from external sources that can trigger interrupts in the microcontroller or microprocessor. It may include functions for configuring and handling external interrupts from external devices, such as buttons or sensors.

**LED** he LED module is responsible for controlling the LEDs (LED1, LED2, LED3, LED4) mentioned in the system requirements. It may include functions for setting the LED states (e.g., ON or OFF), controlling LED brightness or color (if applicable), and handling any other operations related to LED control

**Timer:** This module deals with timer operations, such as configuring and handling timers in the microcontroller or microprocessor. It may include functions for setting timer intervals, handling timer interrupts, and measuring time. And This module deals with generating PWM signals using normal mode, which are used for controlling the intensity of an output signal, such as controlling the speed of motors or the brightness of LEDs. It may include functions for configuring and controlling PWM signals.

**BIT\_MATH**: This module provides functions for performing bitwise operations, such as AND, OR, XOR, and shifting, which are commonly used for manipulating individual bits in registers or memory locations.

**Standard Types:** This module includes standard data types, such as integer types, floating-point types, and Boolean types, which are used for representing data in a standardized way across the system.

# API

## DIO

EN\_dioError\_t DIO\_init(uint8\_t portNumber, uint8\_t pinNumber, uint8\_t direction);

EN\_dioError\_t DIO\_write(uint8\_t portNumber, uint8\_t pinNumber, uint8\_t value);

EN\_dioError\_t DIO\_toggle(uint8\_t portNumber, uint8\_t pinNumber);

EN\_dioError\_t DIO\_read(uint8\_t portNumber, uint8\_t pinNumber, uint8\_t \*value);

## EXTERNAL INTERRUPTS

EN\_extintError\_t EXTINT\_Init (uint8\_t intNumber);

void EXTINT\_setCallBackInt (uint8\_t intNumber, void (\*funPtr) (void));

## LED

EN\_ledError\_t LED\_init(uint8\_t ledPort,uint8\_t ledPin);

EN\_ledError\_t LED\_on(uint8\_t ledPort,uint8\_t ledPin);

EN\_ledError\_t LED\_off(uint8\_t ledPort,uint8\_t ledPin);

EN\_ledError\_t LED\_toggle(uint8\_t ledPort,uint8\_t ledPin);

## BUTTON

EN\_buttonError\_t BUTTON\_init(uint8\_t buttonPort, uint8\_t buttonPin);

EN\_buttonError\_t BUTTON\_read(uint8\_t buttonPort, uint8\_t buttonPin, uint8\_t \*buttonState);

## APP

void APP\_initModules(void);

void APP\_superLoop (void);

void APP\_carMoveForward(void);

void APP\_carTurnRight(void);

void APP\_testingMotors(void);

void APP\_sysTickTask(void);

void APP\_carMoveRight(void);

void APP\_carStop(void);

void APP\_button1Task(void);

## TIMERS

en\_timerError\_t TIMER\_init(u8 u8\_a\_timerUsed);

en\_timerError\_t TIMER\_setTime(u8 u8\_a\_timerUsed, u32 u32\_a\_desiredTime);

en\_timerError\_t TIMER\_start(u8 u8\_a\_timerUsed);

en\_timerError\_t TIMER\_stop(u8 u8\_a\_timerUsed);

en\_timerError\_t TIMER\_pwmGenerator(u8 u8\_a\_timerUsed, u32 u32\_a\_desiredDutyCycle);

void TIMER\_setCallBack(u8 u8\_a\_timerUsed, void (\*funPtr)(void));

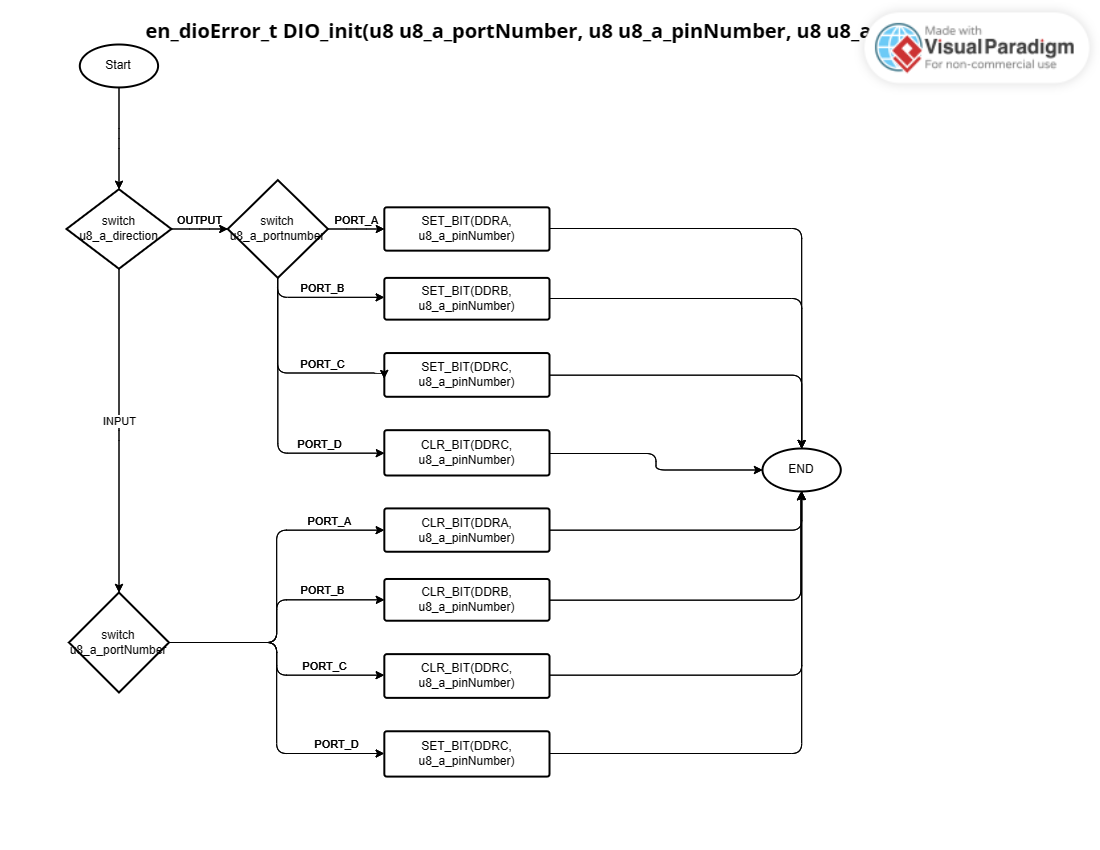
en\_timerError\_t TIMER\_stopInterrupt(u8 u8\_a\_timerUsed);

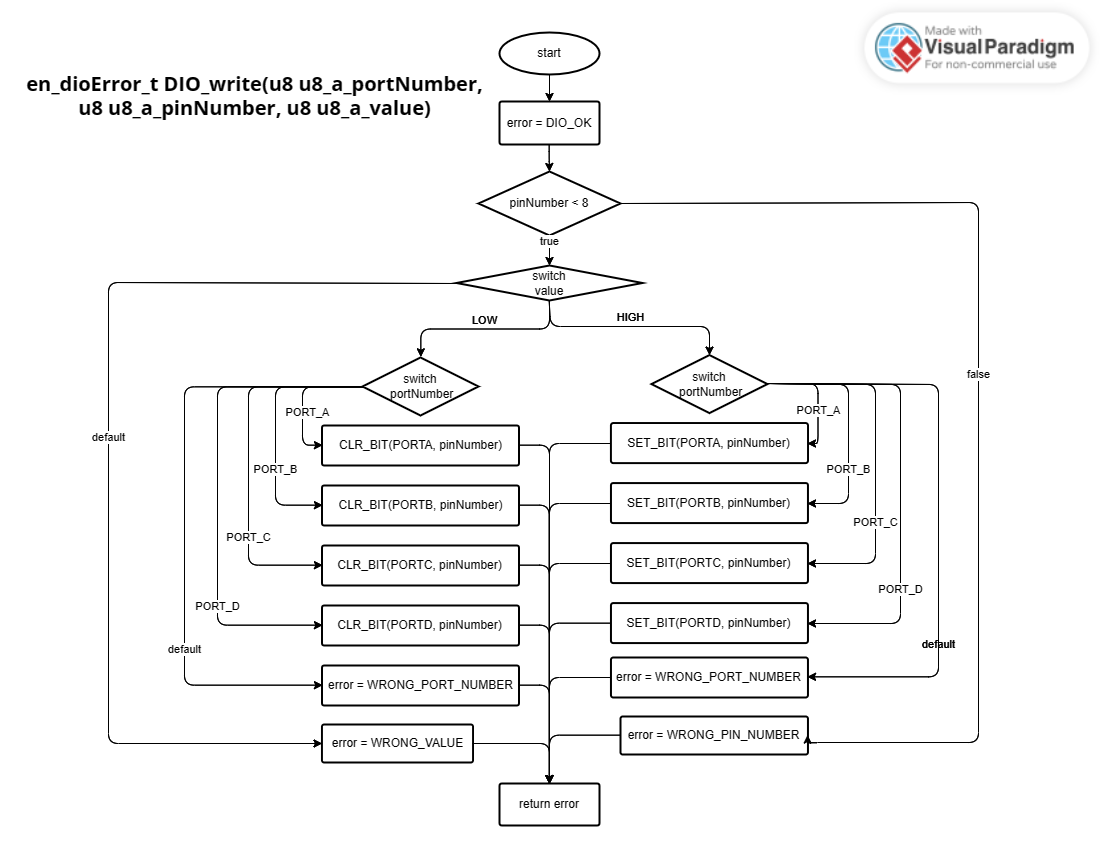
en\_timerError\_t TIMER\_delay(u8 u8\_a\_timerUsed, u32 u32\_a\_timeInMS);

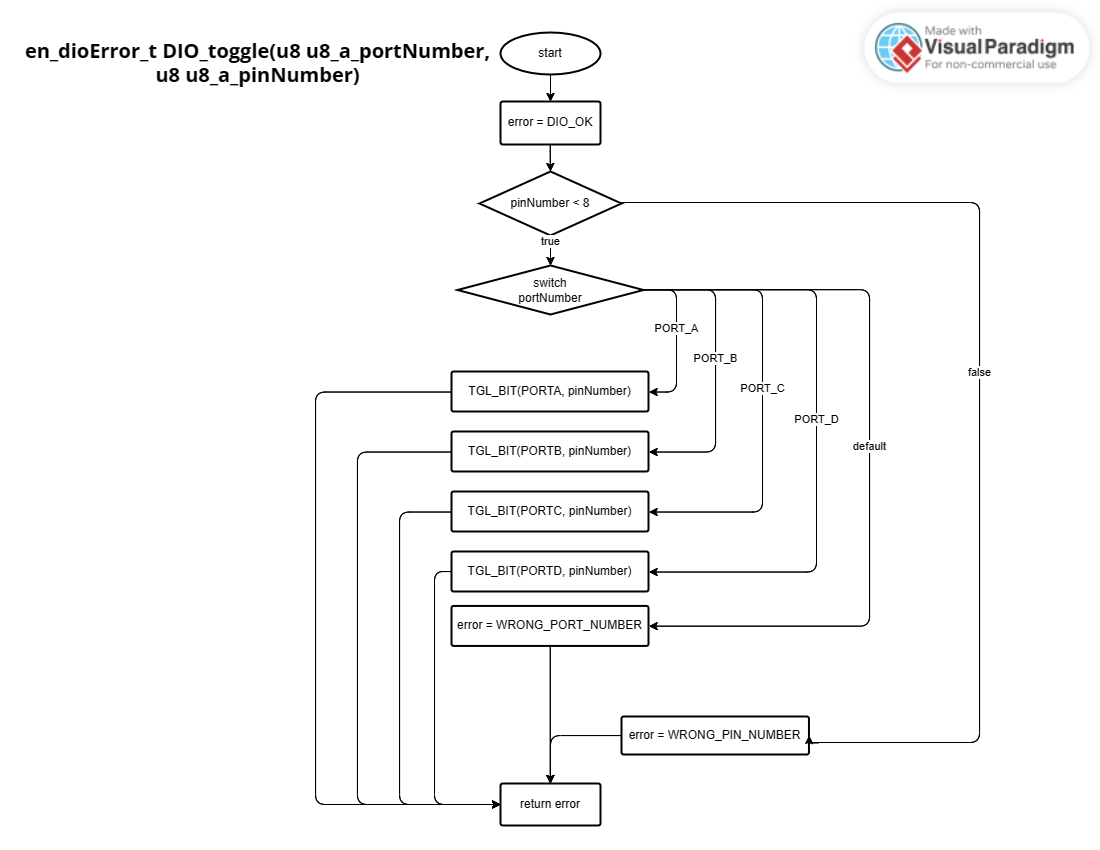
en\_timerError\_t TIMER\_enableInterrupt(u8 u8\_a\_timerUsed);

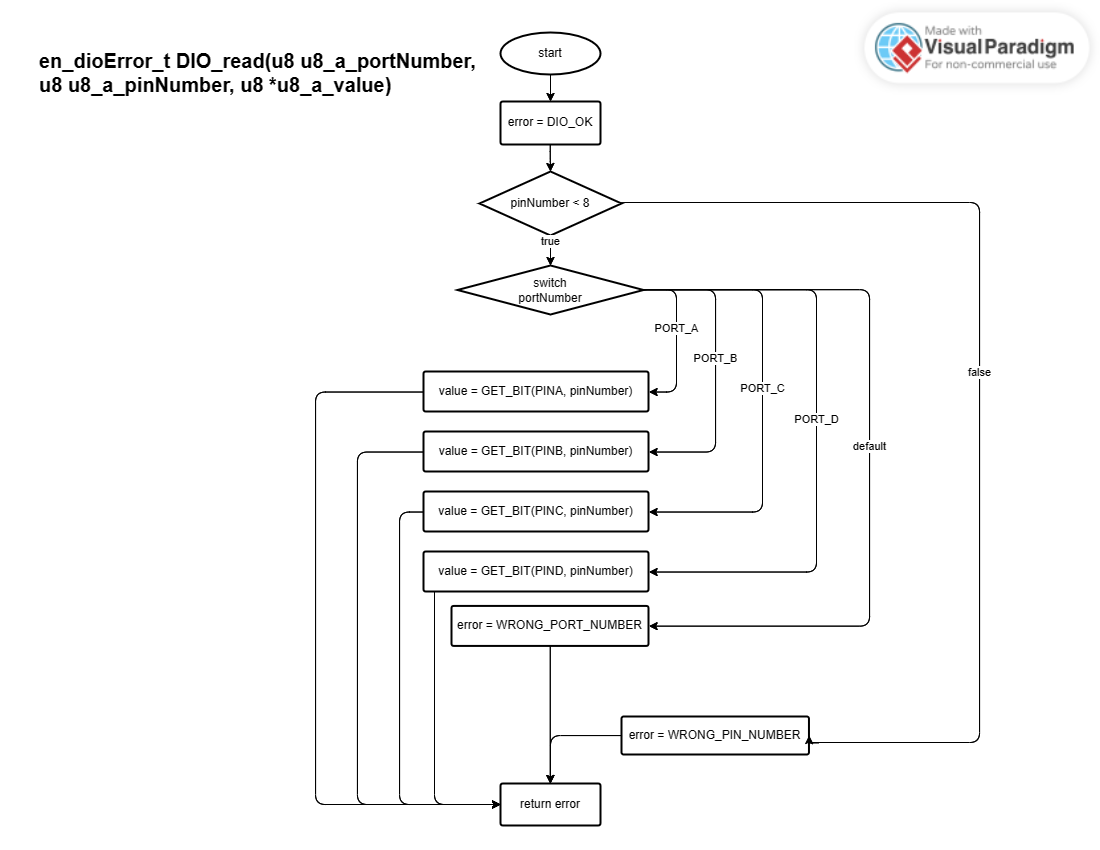
# FUNCTIONS FLOWCHART

## DIO

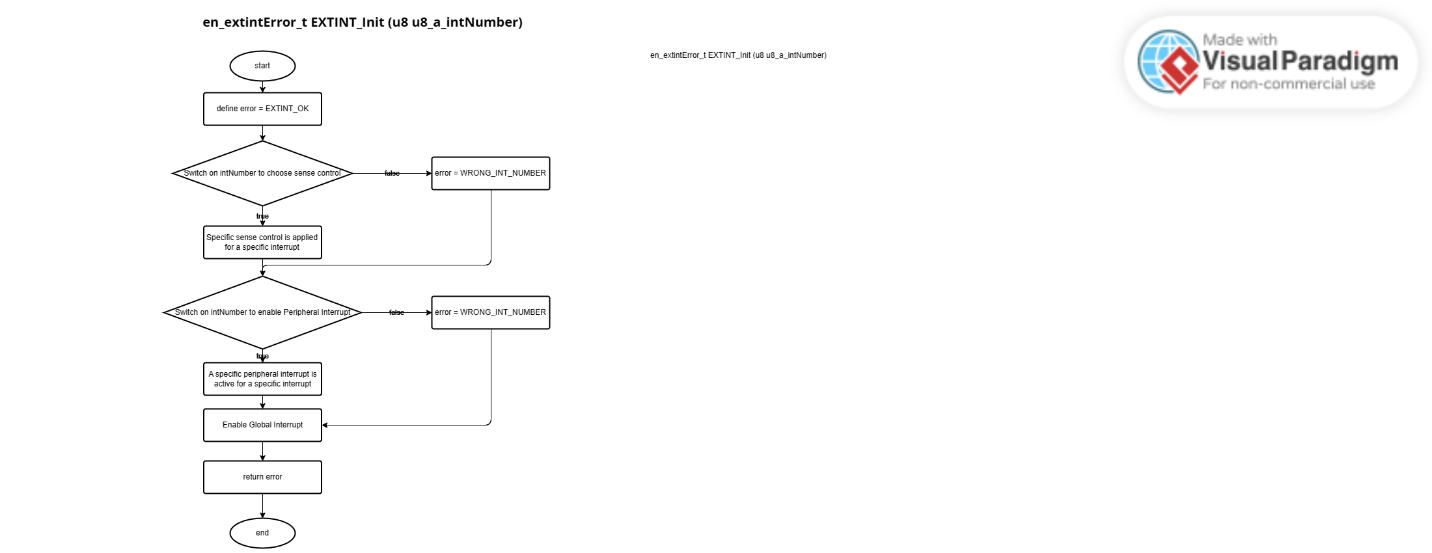


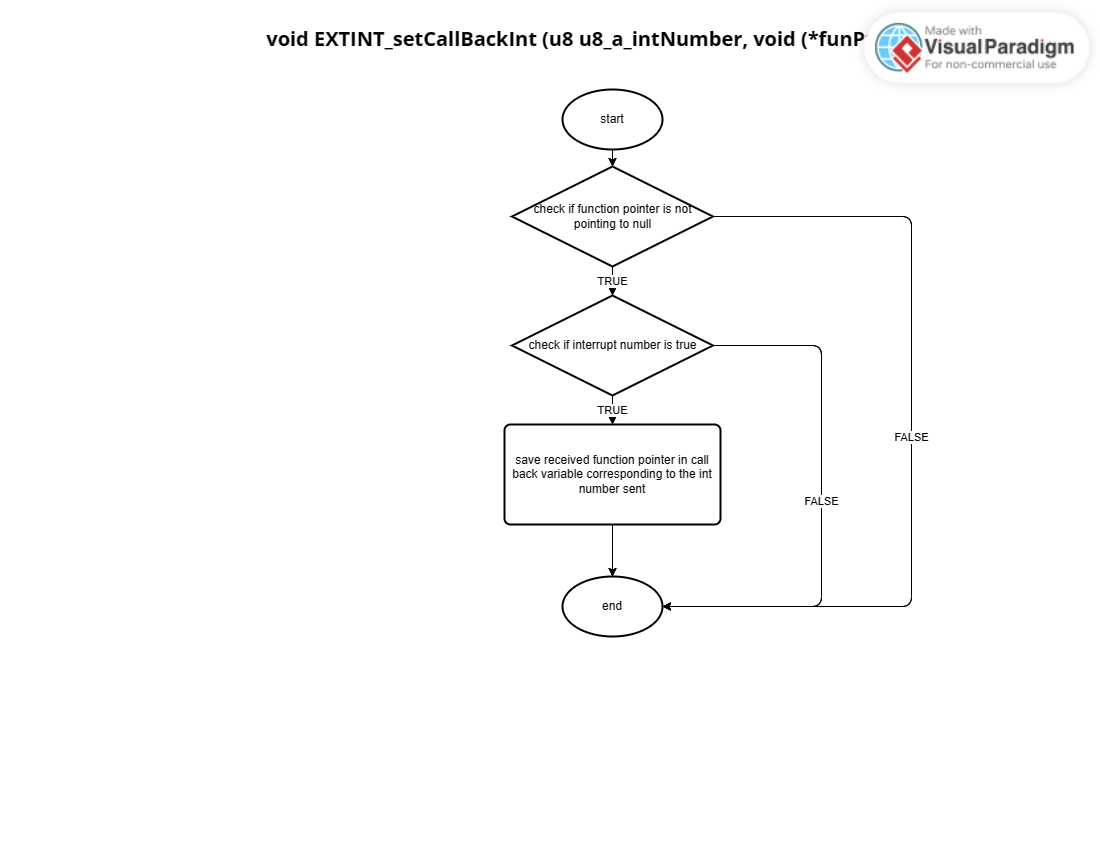




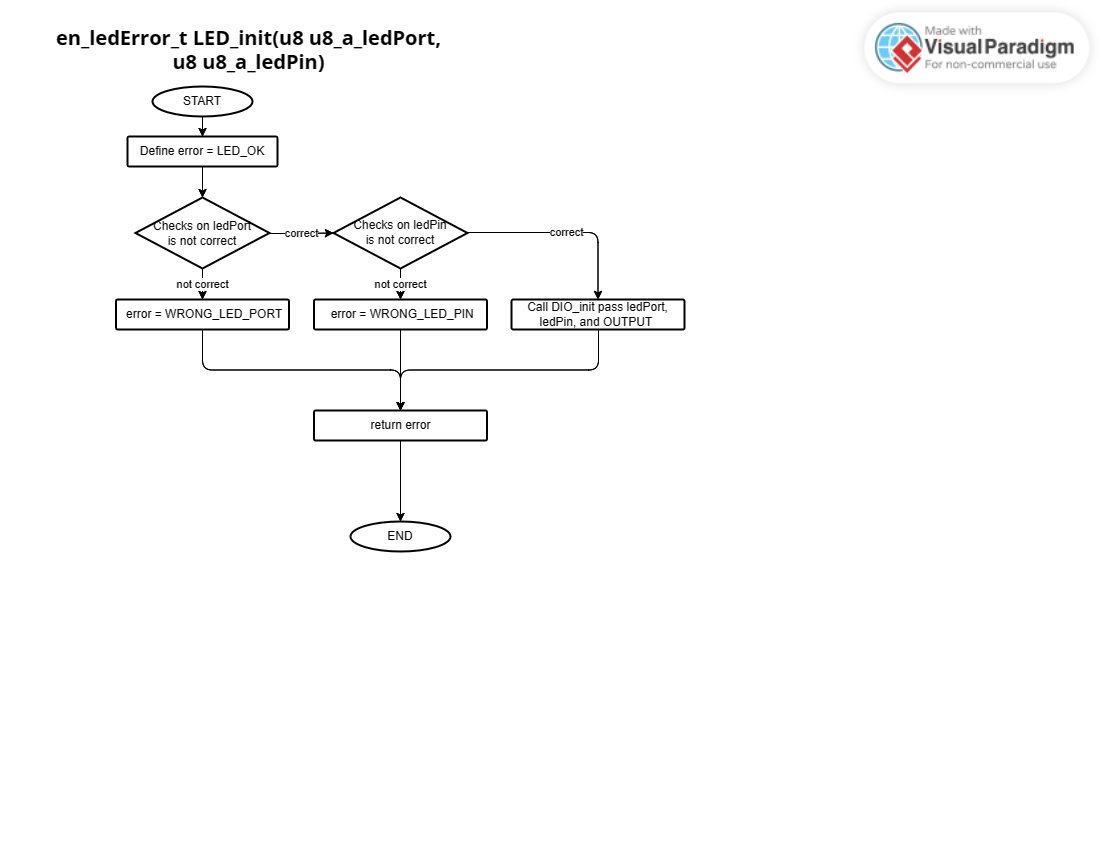


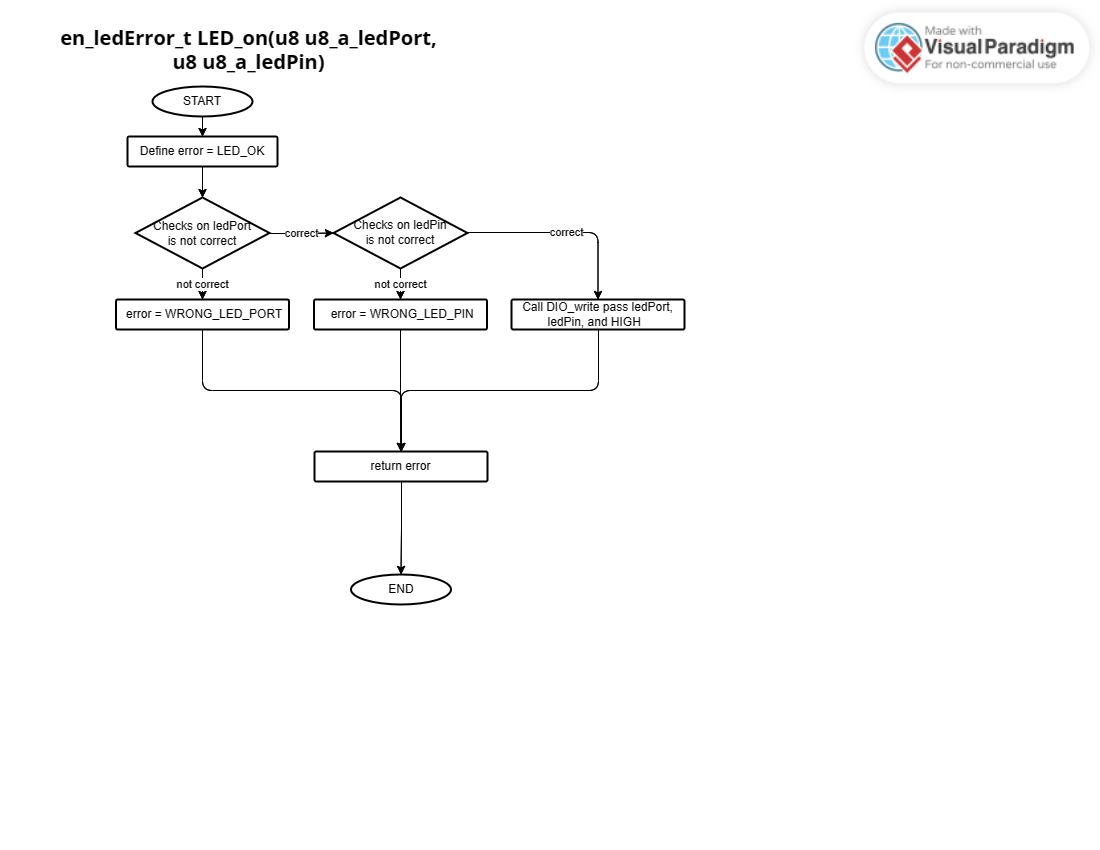
## 2.EXTERNAL INTERRUPTS

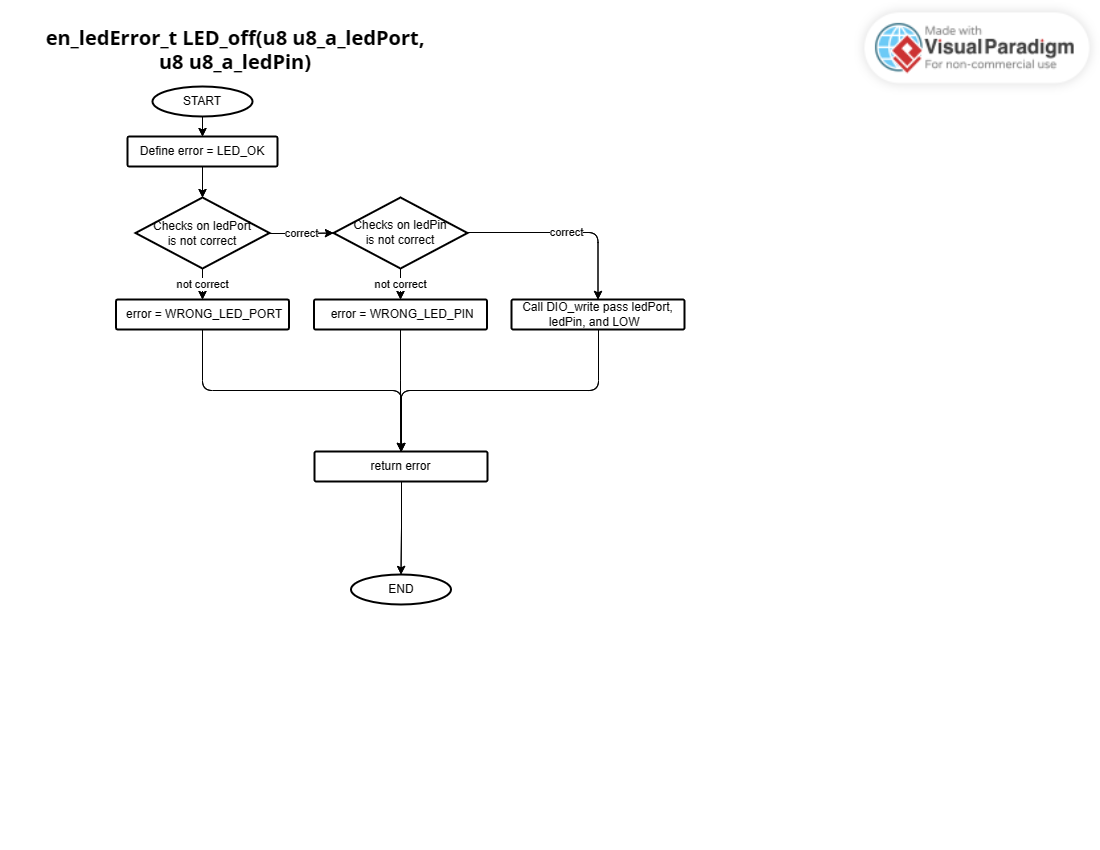


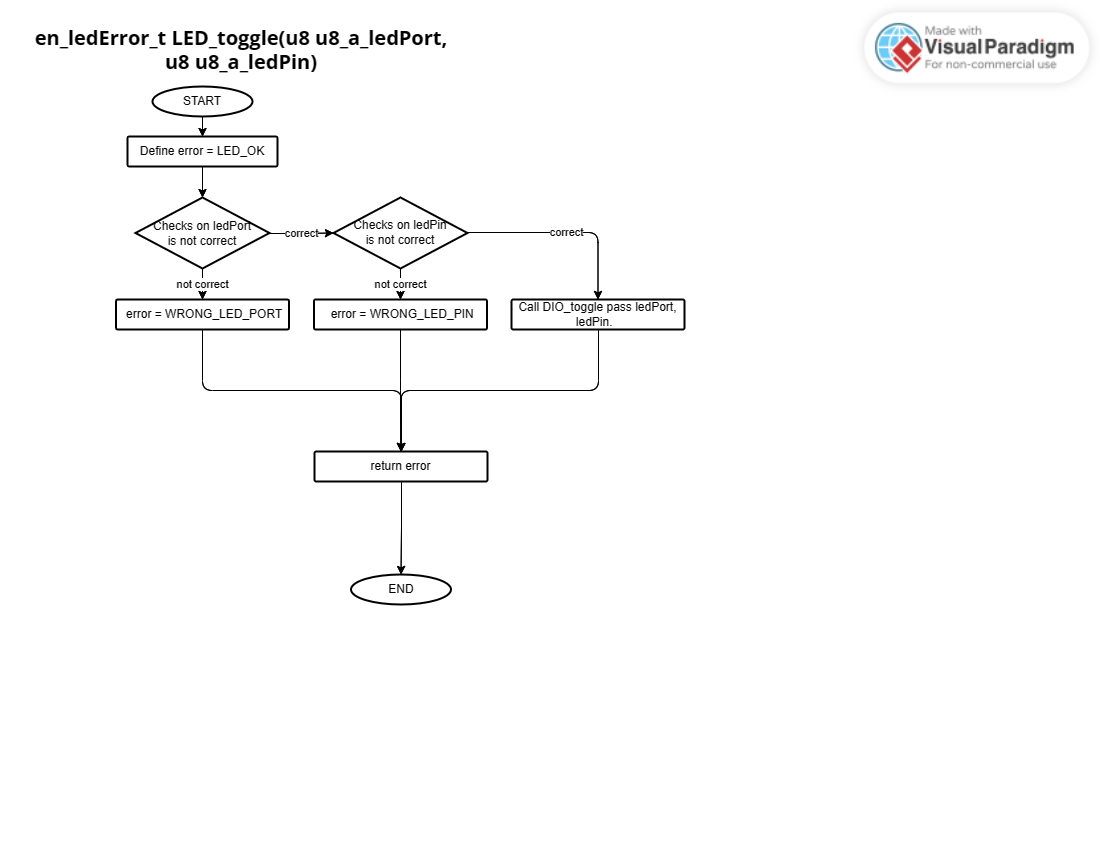


## 3.LED

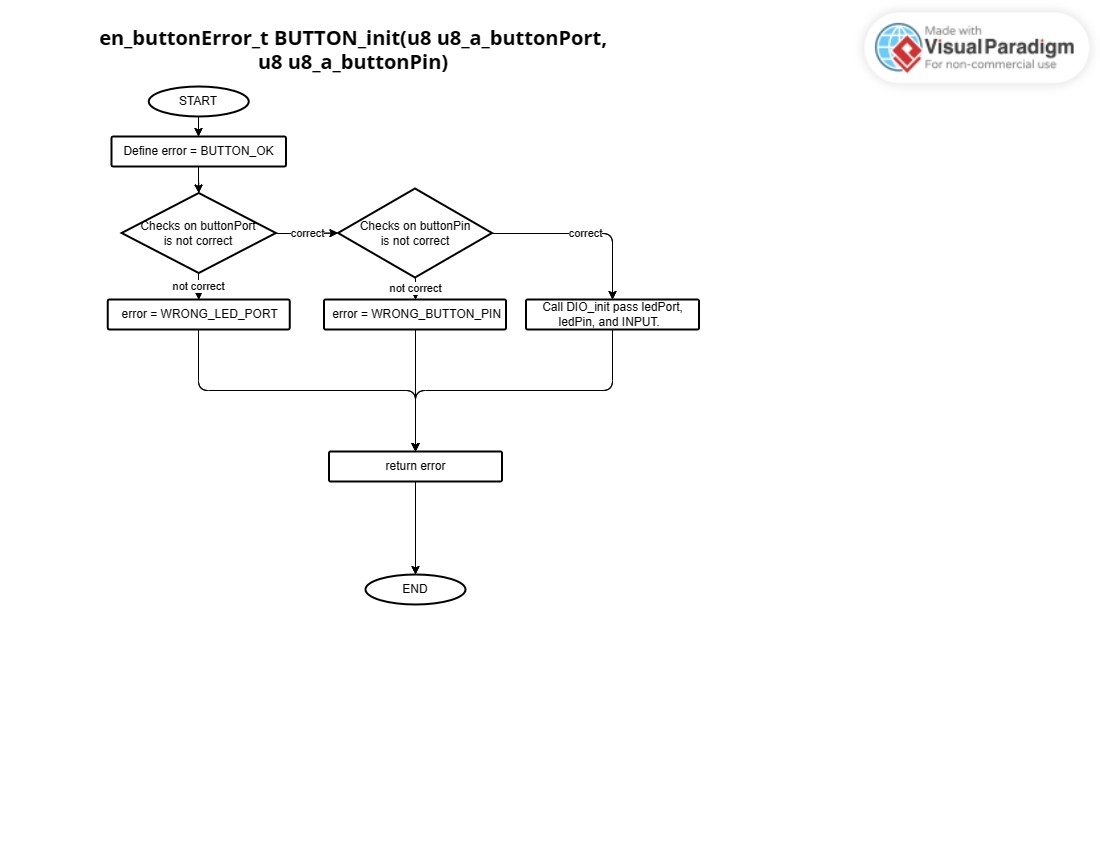


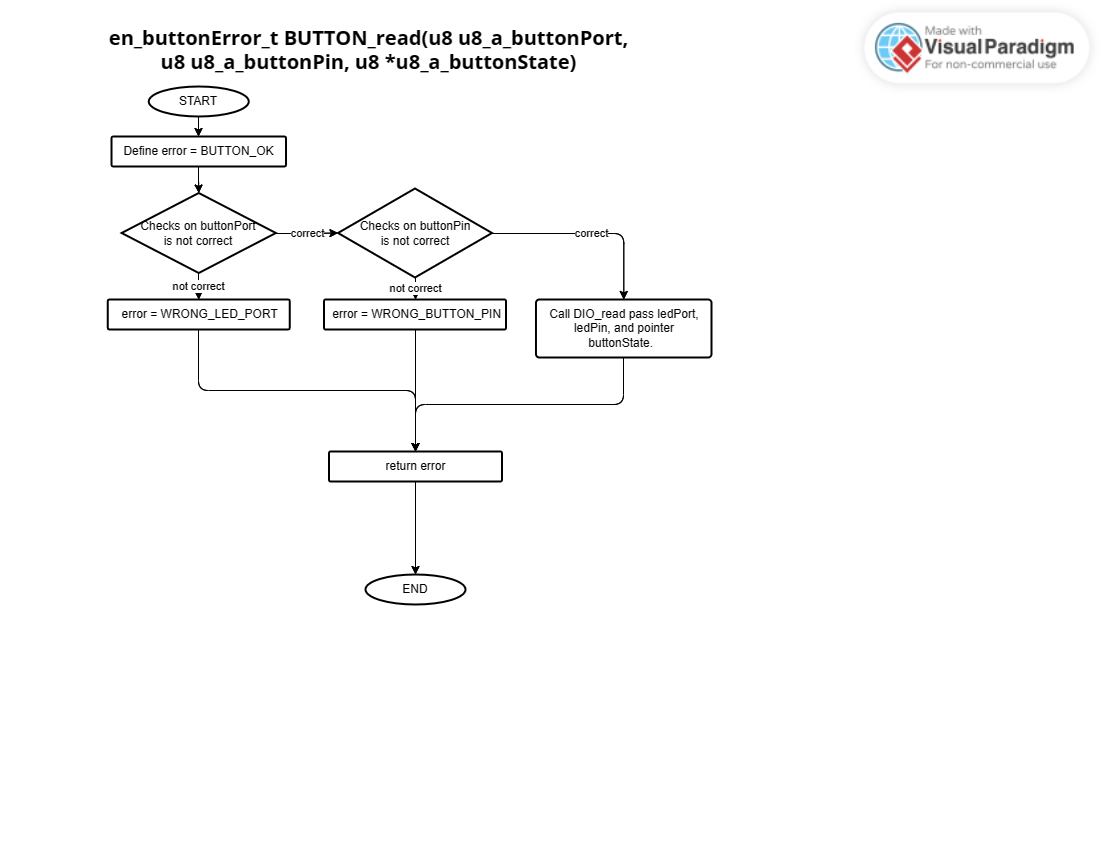






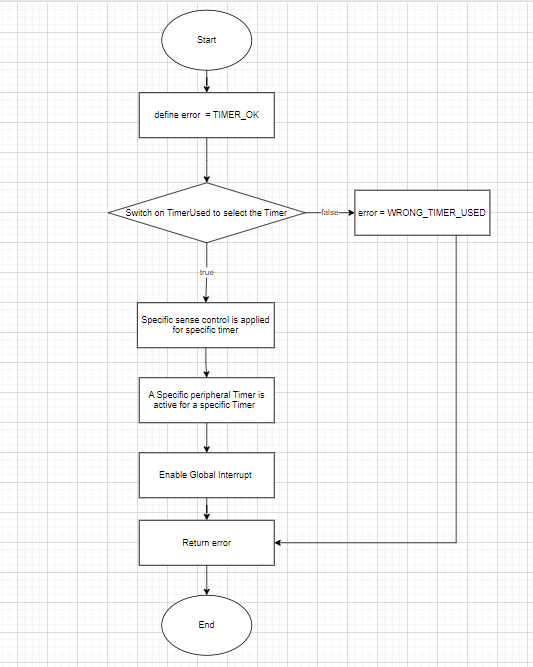
## 4.BUTTON



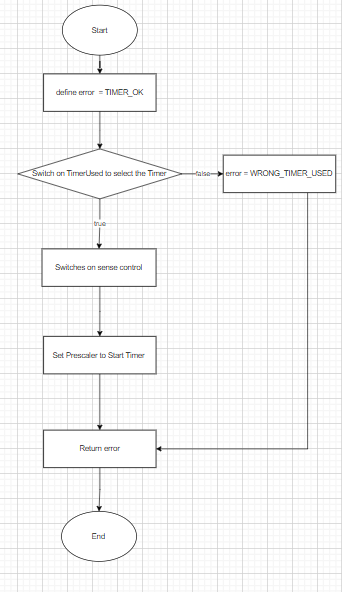


## 5.TIMERS

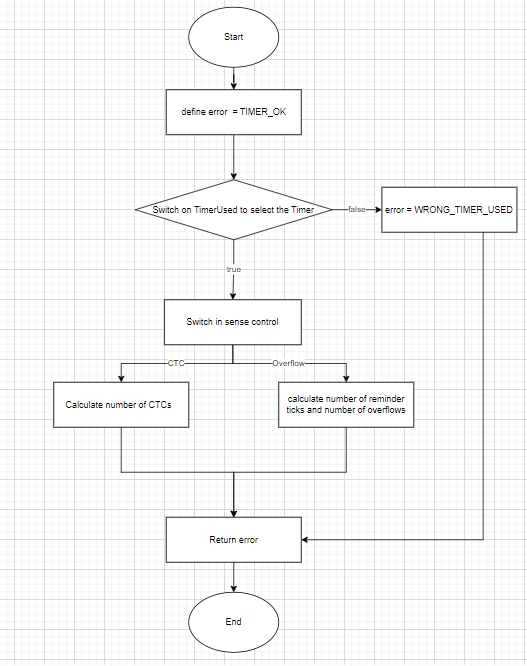
en\_timerError\_t TIMER\_init(u8 u8\_a\_timerUsed);



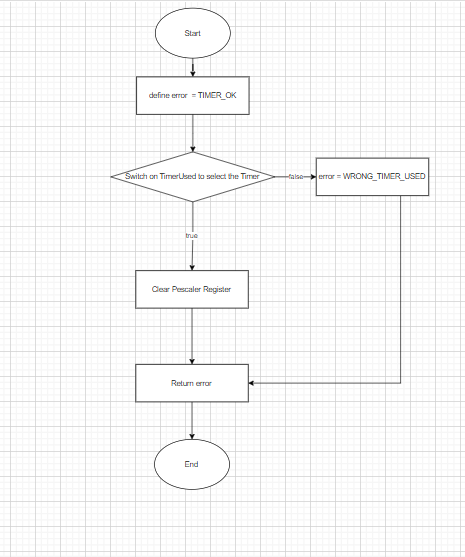
en\_timerError\_t TIMER\_start(u8 u8\_a\_timerUsed);



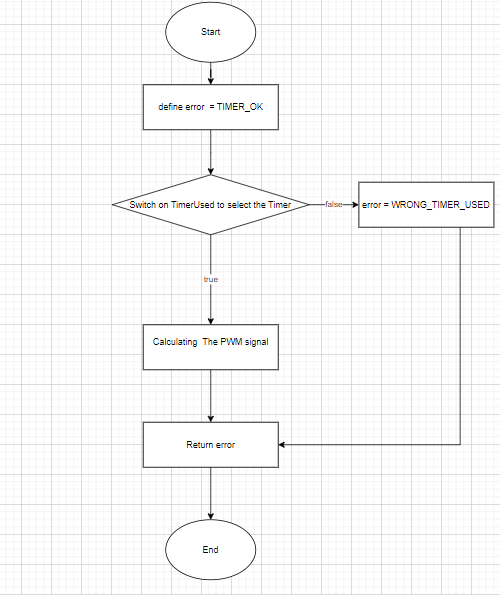
en\_timerError\_t TIMER\_setTime(u8 u8\_a\_timerUsed, u32 u32\_a\_desiredTime);



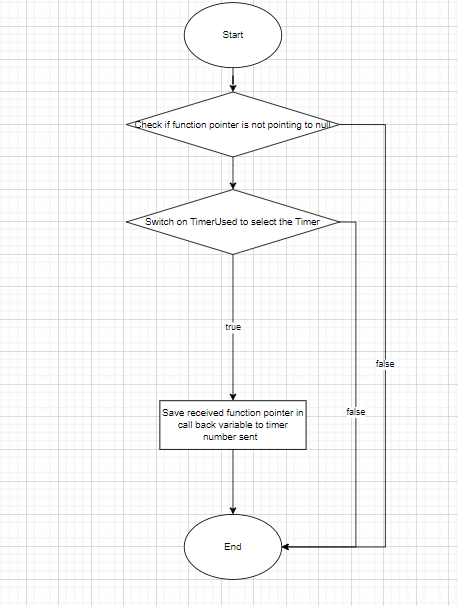
en\_timerError\_t TIMER\_stop(u8 u8\_a\_timerUsed);



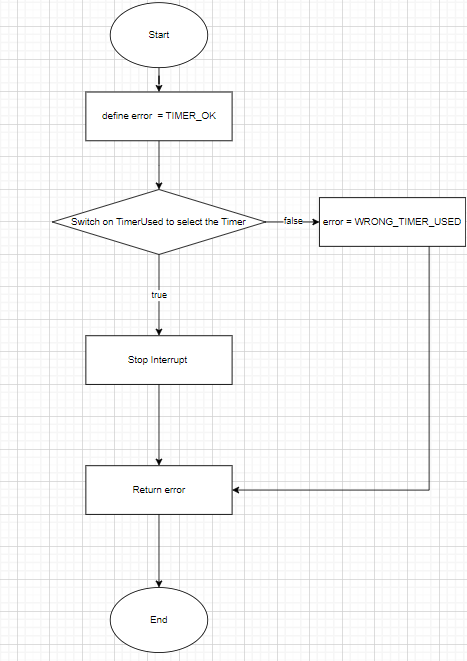
en\_timerError\_t TIMER\_pwmGenerator(u8 u8\_a\_timerUsed, u32 u32\_a\_desiredDutyCycle);



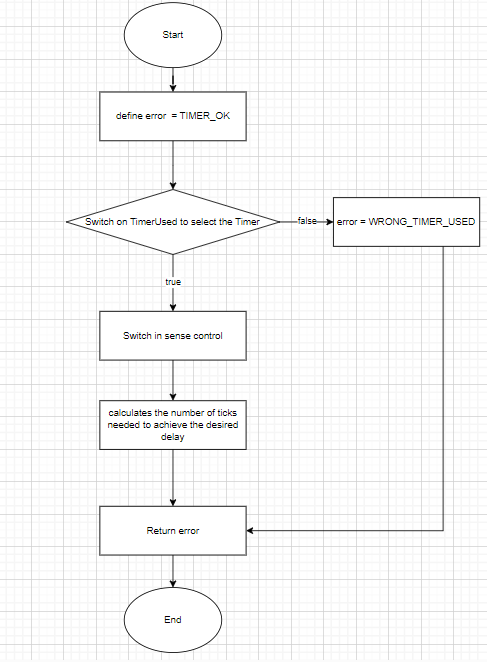
Void TIMER\_setCallBack(u8 u8\_a\_timerUsed, void (\*funPtr)(void));



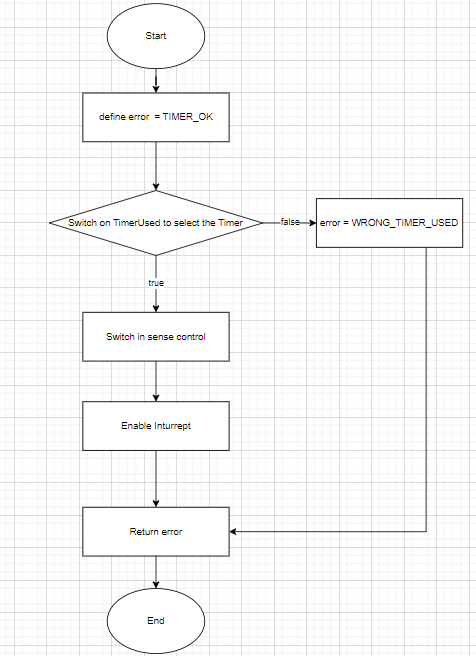
en\_timerError\_t TIMER\_stopInterrupt(u8 u8\_a\_timerUsed);



en\_timerError\_t TIMER\_delay(u8 u8\_a\_timerUsed, u32 u32\_a\_timeInMS);



en\_timerError\_t TIMER\_enableInterrupt(u8 u8\_a\_timerUsed);



## 6.APP

