Design document

project title: RGB LED brightness control V1.0 Design

Represented by: Hazem Ashraf

Team: 3

Table of contents

• Project Description

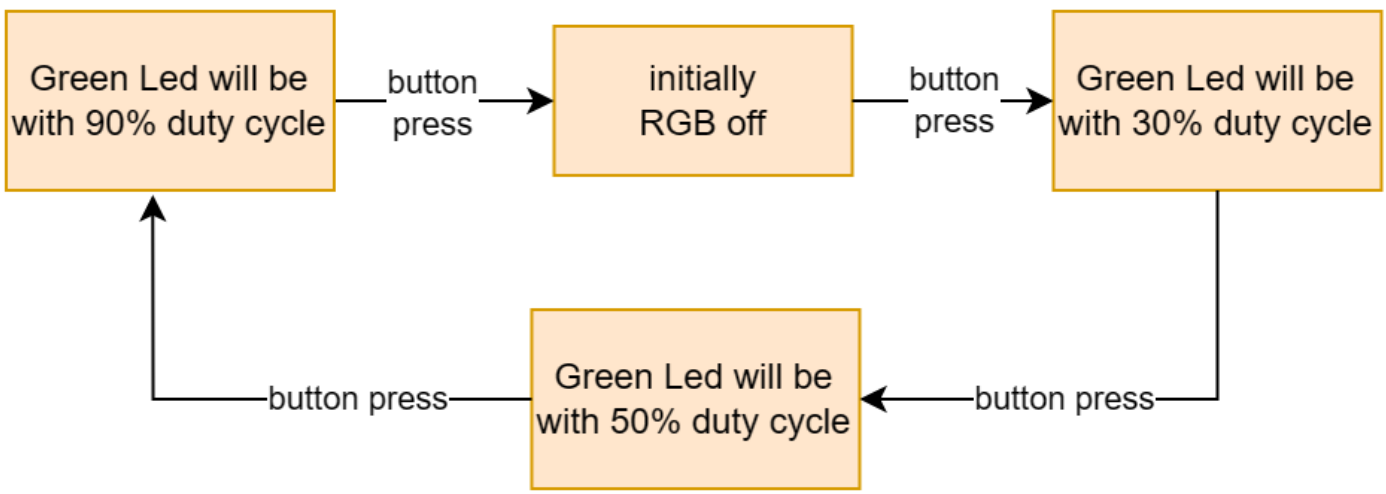
You are supposed to develop the Timer Driver and use it to control the RGB LED brightness on the TivaC board based on the push button press

• Project components

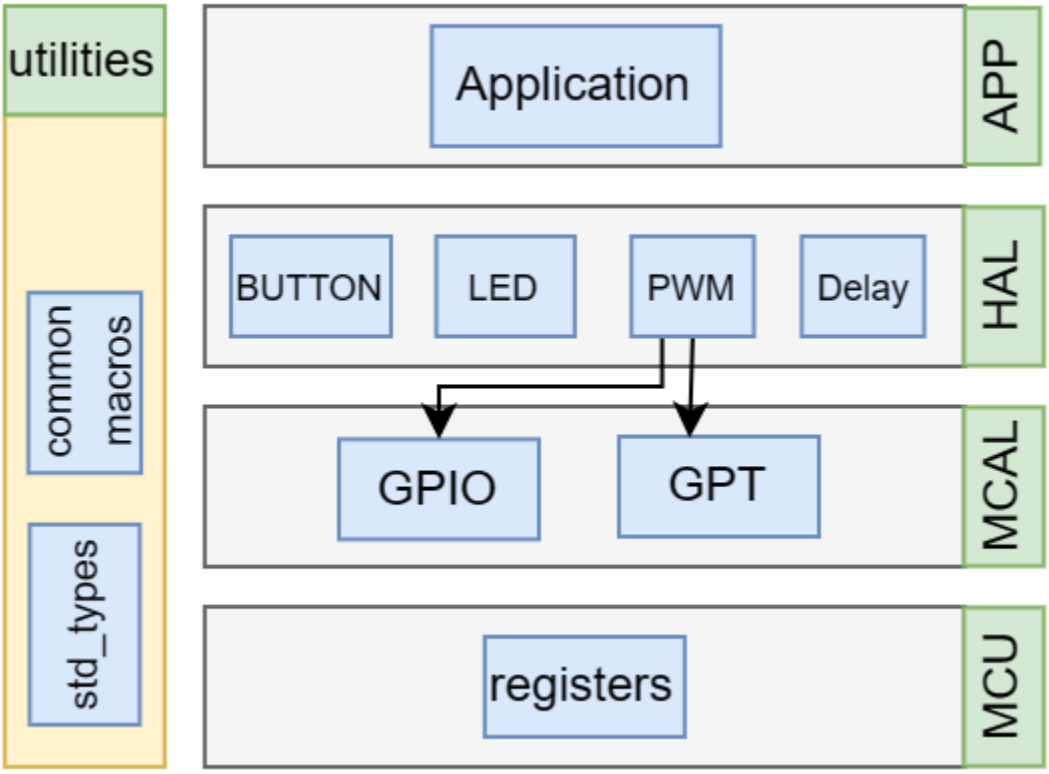
Use the TivaC board
Use SW1 as an input button
Use the RGB LED

• Main Application Flow

The RGB LED is OFF initially
The PWM signal has a 500ms duration
The system has four states
SW1 - First press
The Green LED will be on with a 30% duty cycle
SW1 - Second press
The Green LED will be on with a 60% duty cycle
SW1 -Third press
The Green LED will be on with a 90% duty cycle
SW1 - Fourth press will be off
The Green LED will be off
On the fifth press, system state will return to state 1



- Layered architecture



- **Layer Architecture Description**

  - **Application Layer**

refers to a software layer used for system- and application-specific purposes that is decoupled from the underlying hardware. The application code meets product-specific features and requirements.

  - **Hardware abstraction layer (HAL)**

refers to a firmware layer that replaces hardware-level accesses with higher-level function calls.

  - **MCAL**

refers to the software layer that contains low-level, microcontroller-specific software. The driver layer forms the basis from which higher-level software interacts with and controls the microcontroller.

  - **Utilities**

Refers to the layer that contain system utilities and any software that could be used with any layer

- **Modules Description**

  - **APP Layer**

    Contain the implementation of application initialization and application start

  - **HAL modules**

BUTTON

Used to configure button pin as input and it is used for change LED state

LED

Used to configure LED pin as output and it is used to control LED state

PWM

Used to control RGB Led brightness

Delay

Used during button reading, Pushbuttons often generate spurious open/close transitions when pressed, due to mechanical and physical issues: to Debounce an input we use delay which is the amount of time it takes for a switch to register a key pressor mouse click

  - **MCAL modules**

GPIO

Used to configure pins directions and read the pin if it is direction is input and write high / low if it is directions is output. Using GPIO for initialize BUTTON ,LED

GPT

Used to configure the Timer to act in a dedicated mode Used in implement the PWM and Delay

- **APP APIs**

1- app_init function will initialize the button and led

Function Name	**app_init**
Syntax	**void app_init (void);**
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	None
Parameters (out):	None
Return	None

2- app_start function run while(1) to start program logic

Function Name	**app_start**
Syntax	**void app_start (void);**
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	None
Parameters (out):	None
Return	None

- **GPIO APIs**

1- GPIO_init function will initialize a specific pin with the required configuration

Function Name	**GPIO_init**
Syntax	**enu_gpio_error_state_t GPIO_init (const str_gpio_config_t* str_gpio_config);**
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**str_gpio_config:** pointer to structure of gpio configuration type
Parameters (out):	None
Return	ENU_INVALID : in case invalid configuration parameter ENU_VALID : in case valid configuration parameter

2- GPIO_digitalWrite used to write high/ low to specific pin

Function Name	**GPIO_digitalWrite**
Syntax	**enu_gpio_error_state_t GPIO_digitalWrite (enu_gpio_port_id_t enu_gpio_port_id ,enu_gpio_pin_id_t enu_gpio_pin_id ,enu_gpio_pin_level_t enu_gpio_pin_level);**
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**enu_gpio_port_id :** port id should be one of the following [ENU_PORT_A, ENU_PORT_B, ENU_PORT_C, ENU_PORT_D, ENU_PORT_E, ENU_PORT_F] **enu_gpio_pin_id** :pin id should be one of the following [ENU_PIN_0, ENU_PIN_1, ENU_PIN_2, ENU_PIN_3, ENU_PIN_4, ENU_PIN_5, ENU_PIN_6, ENU_PIN_7] **enu_gpio_pin_level :**the value of the pin ,should be [ENU_PIN_LOW, ENU_PIN_HIGH]
Parameters (out):	None
Return	ENU_INVALID : in case invalid configuration parameter ENU_VALID : in case valid configuration parameter

- **GPIO APIs**

3- GPIO_digitalRead used to read high/ low from specific pin

Function Name	GPIO_digitalRead
Syntax	**enu_gpio_error_state_t GPIO_digitalRead (enu_gpio_port_id_t enu_gpio_port_id,enu_gpio_pin_id_t enu_gpio_pin_id,uint8* P_value);**
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**enu_gpio_port_id :** port id should be one of the following [ENU_PORT_A, ENU_PORT_B, ENU_PORT_C, ENU_PORT_D, ENU_PORT_E, ENU_PORT_F] **enu_gpio_pin_id** :pin id should be one of the following [ENU_PIN_0, ENU_PIN_1, ENU_PIN_2, ENU_PIN_3, ENU_PIN_4, ENU_PIN_5, ENU_PIN_6, ENU_PIN_7]
Parameters (out):	**P_value:** the value of the required pin
Return	ENU_INVALID : in case invalid configuration parameter ENU_VALID : in case valid configuration parameter

4- GPIO_togglePin used to toggle value of specific pin

Function Name	GPIO_togglePin
Syntax	**enu_gpio_error_state_t GPIO_togglePin (enu_gpio_port_id_t enu_gpio_port_id,enu_gpio_pin_id_t enu_gpio_pin_id);**
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**enu_gpio_port_id :** port id should be one of the following [ENU_PORT_A, ENU_PORT_B, ENU_PORT_C, ENU_PORT_D, ENU_PORT_E, ENU_PORT_F] **enu_gpio_pin_id** :pin id should be one of the following [ENU_PIN_0, ENU_PIN_1, ENU_PIN_2, ENU_PIN_3, ENU_PIN_4, ENU_PIN_5, ENU_PIN_6, ENU_PIN_7]
Parameters (out):	**None**
Return	ENU_INVALID : in case invalid configuration parameter ENU_VALID : in case valid configuration parameter

- **GPIO APIs**

5- GPIO_interruptEnable used set or enable interrupt configuration

Function Name	**GPIO_digitalRead**
Syntax	**enu_gpio_error_state_t GPIO_interruptEnable (enu_interrupt_edge_t enu_interrupt_edge,enu_gpio_port_id_t enu_gpio_port_id,enu_gpio_pin_id_t enu_gpio_pin_id);**
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**enu_interrupt_edge :** the interrupt trigger type it should be [ENU_LEVEL, ENU_RISING, ENU_FALLING, ENU_ANY_EDGE_CHANGE] **enu_gpio_port_id :** port id should be one of the following [ENU_PORT_A, ENU_PORT_B, ENU_PORT_C, ENU_PORT_D, ENU_PORT_E, ENU_PORT_F] **enu_gpio_pin_id** :pin id should be one of the following [ENU_PIN_0, ENU_PIN_1, ENU_PIN_2, ENU_PIN_3, ENU_PIN_4, ENU_PIN_5, ENU_PIN_6, ENU_PIN_7]
Parameters (out):	**None**
Return	ENU_INVALID : in case invalid configuration parameter ENU_VALID : in case valid configuration parameter

6- GPIO_interruptDisable used disable interrupt

Function Name	**GPIO_interruptDisable**
Syntax	**void GPIO_interruptDisable (enu_gpio_port_id_t enu_gpio_port_id,enu_gpio_pin_id_t enu_gpio_pin_id);**
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**enu_gpio_port_id :** port id should be one of the following [ENU_PORT_A, ENU_PORT_B, ENU_PORT_C, ENU_PORT_D, ENU_PORT_E, ENU_PORT_F] **enu_gpio_pin_id** :pin id should be one of the following [ENU_PIN_0, ENU_PIN_1, ENU_PIN_2, ENU_PIN_3, ENU_PIN_4, ENU_PIN_5, ENU_PIN_6, ENU_PIN_7]
Parameters (out):	**None**
Return	None

- **GPIO APIs**

7- GPIO_interruptEnable used set or enable interrupt configuration

Function Name	**GPIO_Setcallback**
Syntax	**enu_gpio_error_state_t GPIO_Setcallback (void (*Fptr)(void), enu_gpio_port_id_t enu_gpio_port_id,enu_gpio_pin_id_t enu_gpio_pin_id);**
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**Fptr :** pointer to the callback function **enu_gpio_port_id :** port id should be one of the following [ENU_PORT_A, ENU_PORT_B, ENU_PORT_C, ENU_PORT_D, ENU_PORT_E, ENU_PORT_F] **enu_gpio_pin_id** :pin id should be one of the following [ENU_PIN_0, ENU_PIN_1, ENU_PIN_2, ENU_PIN_3, ENU_PIN_4, ENU_PIN_5, ENU_PIN_6, ENU_PIN_7]
Parameters (out):	**None**
Return	ENU_INVALID : in case invalid configuration parameter ENU_VALID : in case valid configuration parameter

- **GPT APIs**

1- gpt_Init : used to initialize GPT module to specific mode

Function Name	**gpt_Init**
Syntax	**enu_timer_error_t gpt_Init (str_gpt_config_t* str_gpt_config);**
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**str_gpt_config :** pointer to structure configuration
Parameters (out):	**None**
Return	GPT_INVALID_OPERATION: in case invalid configuration parameter GPT_VALID_OPERATION: in case valid configuration parameter

2- gpt_startTimer : used to start the timer

Function Name	**gpt_startTimer**
Syntax	**enu_timer_error_t gpt_startTimer (enu_timer_id_t enu_timer_id, uint32 u32_time, enu_tick_unit_t enu_tick_unit);**
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**enu_timer_id:** Timer ID should be [TIMER0_ID,TIMER1_ID,TIMER2_ID,TIMER3_ID,TIMER4_ID,TIMER5_ID,TIMER6_ID,TIMER7_ID,TIMER8_ID,TIMER9_ID,TIMER10_ID,TIMER11_ID] **u32_time:** timer preload value **enu_tick_unit:** time unit it should be [USEC,MSEC,SEC]
Parameters (out):	**None**
Return	GPT_INVALID_OPERATION: in case invalid configuration parameter GPT_VALID_OPERATION: in case valid configuration parameter

- **GPT APIs**

3- gpt_enable_notification : used to enable timer interrupt notifications

Function Name	**gpt_enable_notification**
Syntax	**enu_timer_error_t gpt_enable_notification (enu_timer_id_t enu_timer_id)**
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**enu_timer_id:** Timer ID should be [TIMER0_ID,TIMER1_ID,TIMER2_ID,TIMER3_ID,TIMER4_ID,TIMER5_ID,TIMER6_ID,TIMER7_ID,TIMER8_ID,TIMER9_ID,TIMER10_ID,TIMER11_ID]
Parameters (out):	**None**
Return	GPT_INVALID_OPERATION: in case invalid configuration parameter GPT_VALID_OPERATION: in case valid configuration parameter

4- gpt_disable_notification : used to disable timer interrupt notifications

Function Name	**gpt_disable_notification**
Syntax	**enu_timer_error_t gpt_disable_notification (enu_timer_id_t enu_timer_id)**
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**enu_timer_id:** Timer ID should be [TIMER0_ID,TIMER1_ID,TIMER2_ID,TIMER3_ID,TIMER4_ID,TIMER5_ID,TIMER6_ID,TIMER7_ID,TIMER8_ID,TIMER9_ID,TIMER10_ID,TIMER11_ID]
Parameters (out):	**None**
Return	GPT_INVALID_OPERATION: in case invalid configuration parameter GPT_VALID_OPERATION: in case valid configuration parameter

- **GPT APIs**

5- gpt_stopTimer : used to stop the timer

Function Name	**gpt_stopTimer**
Syntax	**enu_timer_error_t gpt_stopTimer (enu_timer_id_t enu_timer_id);**
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**enu_timer_id:** Timer ID should be [TIMER0_ID,TIMER1_ID,TIMER2_ID,TIMER3_ID,TIMER4_ID,TIMER5_ID,TIMER6_ID,TIMER7_ID,TIMER8_ID,TIMER9_ID,TIMER10_ID,TIMER11_ID]
Parameters (out):	**None**
Return	GPT_INVALID_OPERATION: in case invalid configuration parameter GPT_VALID_OPERATION: in case valid configuration parameter

- **LED APIs**

1- LED_init function will initialize LED

Function Name	LED_init
Syntax	enu_error_state_t LED_init (enu_gpio_port_id_t enu_gpio_port_id,enu_gpio_pin_id_t enu_gpio_pin_id);
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**enu_gpio_port_id :** port id should be one of the following [ENU_PORT_A, ENU_PORT_B, ENU_PORT_C, ENU_PORT_D, ENU_PORT_E, ENU_PORT_F] **enu_gpio_pin_id** :pin id should be one of the following [ENU_PIN_0, ENU_PIN_1, ENU_PIN_2, ENU_PIN_3, ENU_PIN_4, ENU_PIN_5, ENU_PIN_6, ENU_PIN_7]
Parameters (out):	None
Return	ENU_INVALID : in case invalid passing parameter ENU_VALID : in case valid passing parameter

2- LED_digitalWrite used to write high/ low to specific LED

Function Name	LED_digitalWrite
Syntax	enu_error_state_t LED_digitalWrite (enu_gpio_port_id_t enu_gpio_port_id,enu_gpio_pin_id_t enu_gpio_pin_id,enu_gpio_pin_level_t enu_gpio_pin_level);
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**enu_gpio_port_id :** port id should be one of the following [ENU_PORT_A, ENU_PORT_B, ENU_PORT_C, ENU_PORT_D, ENU_PORT_E, ENU_PORT_F] **enu_gpio_pin_id** :pin id should be one of the following [ENU_PIN_0, ENU_PIN_1, ENU_PIN_2, ENU_PIN_3, ENU_PIN_4, ENU_PIN_5, ENU_PIN_6, ENU_PIN_7] **enu_gpio_pin_level :**the value of the pin ,should be [ENU_PIN_LOW, ENU_PIN_HIGH]
Parameters (out):	None
Return	ENU_INVALID : in case invalid configuration parameter ENU_VALID : in case valid configuration parameter

- **BUTTON APIs**

1- BUTTON_init function will initialize button

Function Name	BUTTON_init
Syntax	enu_error_state_t BUTTON_init (enu_gpio_port_id_t enu_gpio_port_id,enu_gpio_pin_id_t enu_gpio_pin_id);
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**enu_gpio_port_id :** port id should be one of the following [ENU_PORT_A, ENU_PORT_B, ENU_PORT_C, ENU_PORT_D, ENU_PORT_E, ENU_PORT_F] **enu_gpio_pin_id** :pin id should be one of the following [ENU_PIN_0, ENU_PIN_1, ENU_PIN_2, ENU_PIN_3, ENU_PIN_4, ENU_PIN_5, ENU_PIN_6, ENU_PIN_7]
Parameters (out):	None
Return	ENU_INVALID : in case invalid passing parameter ENU_VALID : in case valid passing parameter

2- BUTTON_digitalRead used to write high/ low to specific LED

Function Name	BUTTON_digitalRead
Syntax	enu_error_state_t BUTTON_digitalRead (enu_gpio_port_id_t enu_gpio_port_id,enu_gpio_pin_id_t enu_gpio_pin_id,uint8* p_value);
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**enu_gpio_port_id :** port id should be one of the following [ENU_PORT_A, ENU_PORT_B, ENU_PORT_C, ENU_PORT_D, ENU_PORT_E, ENU_PORT_F] **enu_gpio_pin_id** :pin id should be one of the following [ENU_PIN_0, ENU_PIN_1, ENU_PIN_2, ENU_PIN_3, ENU_PIN_4, ENU_PIN_5, ENU_PIN_6, ENU_PIN_7]
Parameters (out):	**P_value:** the value of the required pin
Return	ENU_INVALID : in case invalid configuration parameter ENU_VALID : in case valid configuration parameter

- **PWM APIs**

1- PWM_Init :Function used to initialize PWM module

Function Name	PWM_Init
Syntax	**enu_pwm_error_t PWM_Init (enu_gpio_port_id_t enu_gpio_port_id, enu_gpio_pin_id_t enu_gpio_pin_id, enu_timer_id_t enu_timer_id);**
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**enu_gpio_port_id**: port Name ID should Be [ENU_PORT_A,ENU_PORT_B,ENU_PORT_C,ENU_PORT_D,ENU_PORT_E,ENU_PORT_F] **enu_gpio_pin_id**: pin name ID should be [ENU_PIN_0,ENU_PIN_1,ENU_PIN_2,ENU_PIN_3,ENU_PIN_4,ENU_PIN_5,ENU_PIN_6,ENU_PIN_7] **enu_timer_id**: Timer ID should be [TIMER0_ID,TIMER1_ID,TIMER2_ID,TIMER3_ID,TIMER4_ID,TIMER5_ID,TIMER6_ID,TIMER7_ID,TIMER8_ID,TIMER9_ID,TIMER10_ID,TIMER11_ID]
Parameters (out):	None
Return	INVALID : in case invalid passing parameter VALID : in case valid passing parameter

2- PWM_start : Function used to start generate PWM signal

Function Name	PWM_start
Syntax	**enu_pwm_error_t PWM_start (enu_timer_id_t enu_timer_id ,uint32 u32_periodic_time,enu_tick_unit_t enu_tick_unit,uint8 u8_duty_cycle);**
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**enu_timer_id**: Timer ID should be [TIMER0_ID,TIMER1_ID,TIMER2_ID,TIMER3_ID,TIMER4_ID,TIMER5_ID,TIMER6_ID,TIMER7_ID,TIMER8_ID,TIMER9_ID,TIMER10_ID,TIMER11_ID] **u32_periodic_time**: PWM periodic time **enu_tick_unit**: time unit it should be [USEC,MSEC,SEC] **u8_duty_cycle**: PWM duty cycle percentage it should be [0%--100%]
Parameters (out):	None
Return	INVALID : in case invalid passing parameter VALID : in case valid passing parameter

- **PWM APIs**

3- PWM_stop : Function used to stop generate PWM signal

Function Name	**PWM_stop**
Syntax	**enu_pwm_error_t PWM_stop (enu_timer_id_t enu_timer_id);**
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**enu_gpio_port_id**: port name ID should be [ENU_PORT_A,ENU_PORT_B,ENU_PORT_C,ENU_PORT_D,ENU_PORT_E,ENU_PORT_F]
Parameters (out):	None
Return	INVALID : in case invalid passing parameter VALID : in case valid passing parameter

- **Delay APIs**

1- **delay_us** function will make delay in microsecond

Function Name	**delay_us**
Syntax	**enu_delay_error_t delay_us (enu_timer_id_t enu_timer_id ,uint32 u32_time)**
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**enu_timer_id :**Timer ID should be [TIMER0_ID,TIMER1_ID,TIMER2_ID,TIMER3_ID,TIMER4_ID,TIMER5_ID,TIMER6_ID,TIMER7_ID,TIMER8_ID,TIMER9_ID,TIMER10_ID,TIMER11_ID] **u32_time:** delay time in microsecond
Parameters (out):	None
Return	INVALID_OPERATION: in case invalid passing parameter VALID_OPERATION: in case valid passing parameter

2- **delay_ms** function will make delay in millisecond

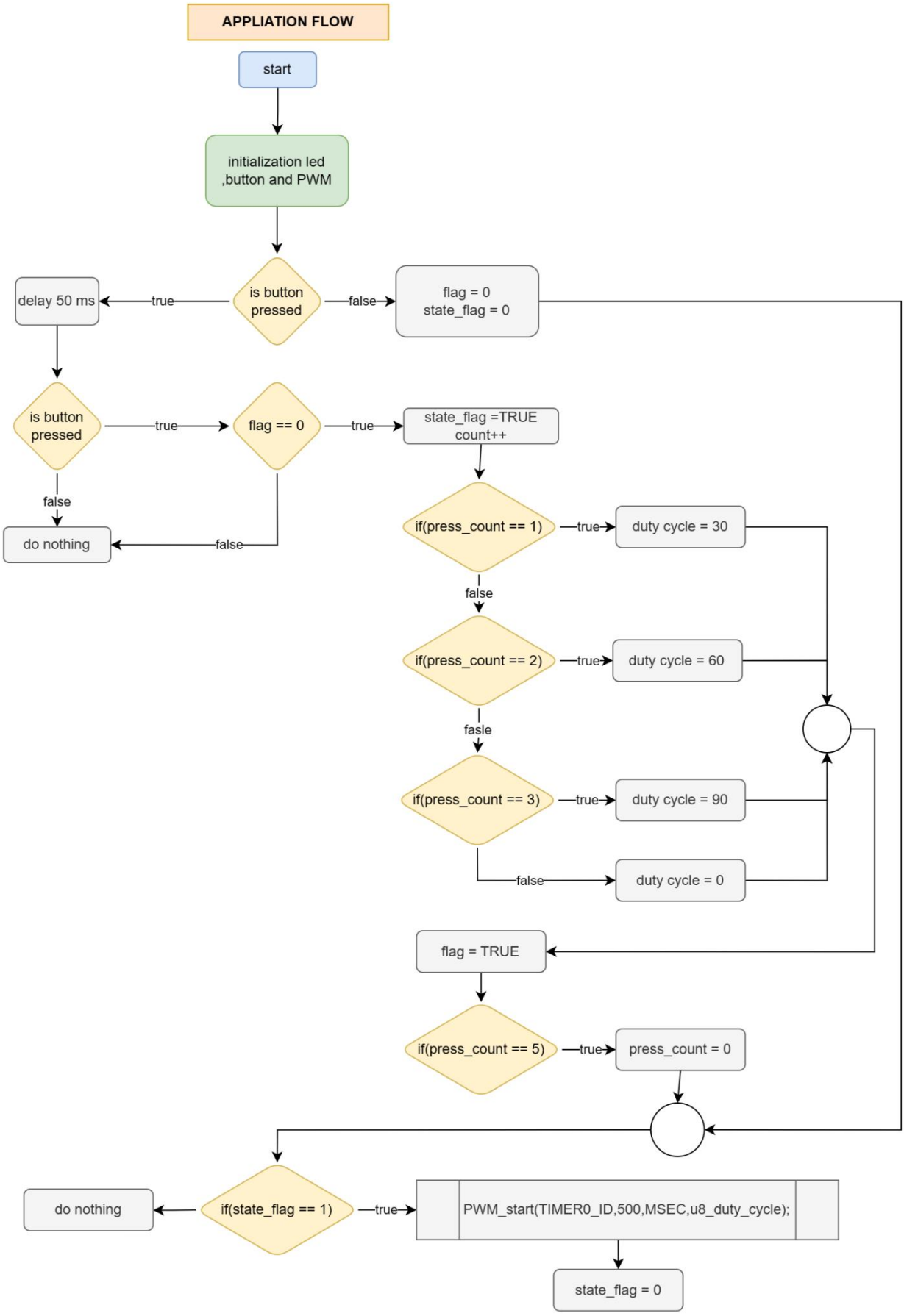
Function Name	**delay_ms**
Syntax	**enu_delay_error_t delay_ms (enu_timer_id_t enu_timer_id ,uint32 u32_time)**
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**enu_timer_id :**Timer ID should be [TIMER0_ID,TIMER1_ID,TIMER2_ID,TIMER3_ID,TIMER4_ID,TIMER5_ID,TIMER6_ID,TIMER7_ID,TIMER8_ID,TIMER9_ID,TIMER10_ID,TIMER11_ID] **u32_time:** delay time in millisecond
Parameters (out):	None
Return	INVALID_OPERATION: in case invalid passing parameter VALID_OPERATION: in case valid passing parameter

- **Delay APIs**

3- **delay_sec** function will make delay in second

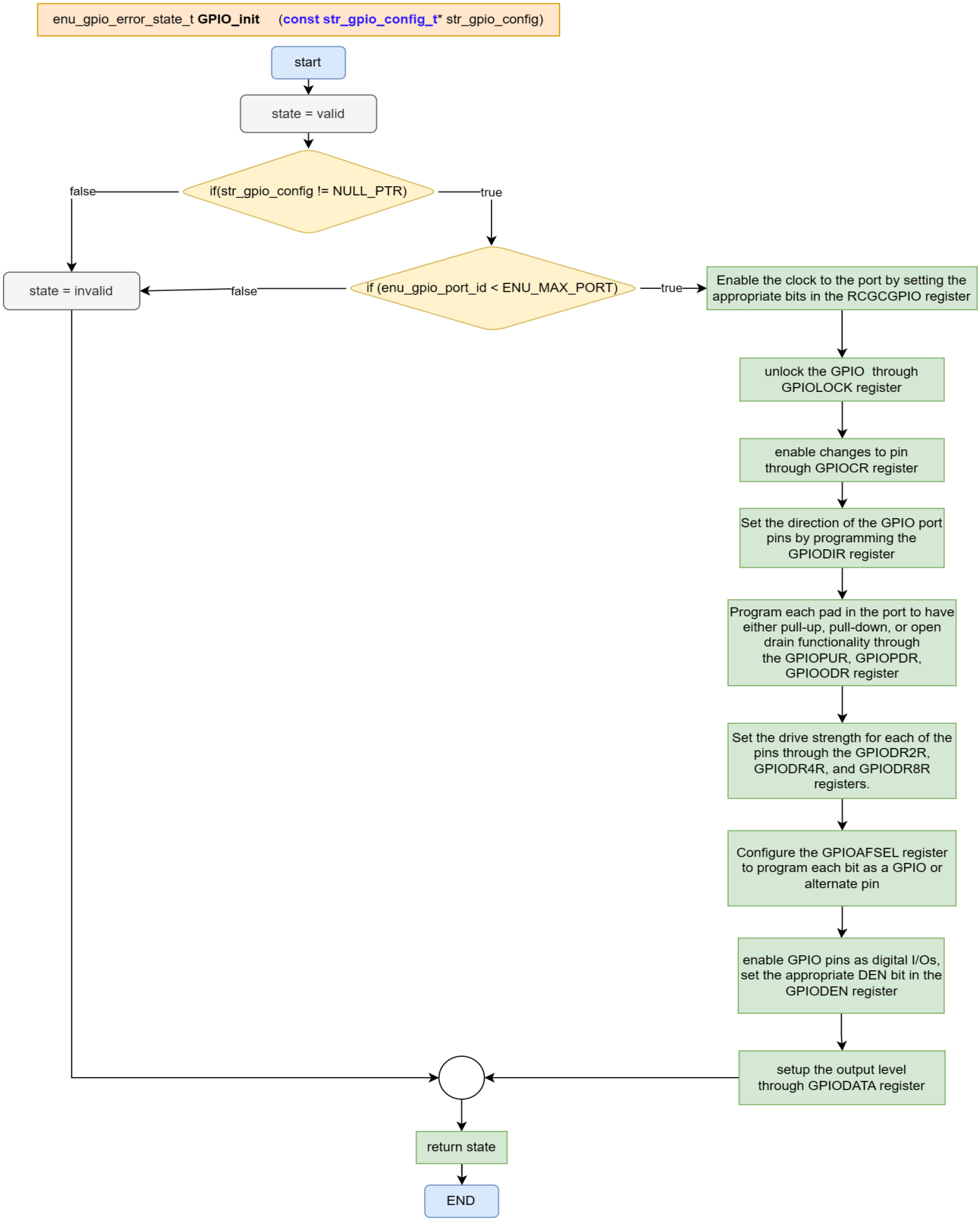
Function Name	**delay_sec**
Syntax	**enu_delay_error_t delay_sec (enu_timer_id_t enu_timer_id ,uint32 u32_time)**
Sync / Asynch.	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**enu_timer_id :**Timer ID should be [TIMER0_ID,TIMER1_ID,TIMER2_ID,TIMER3_ID,TIMER4_ID,TIMER5_ID,TIMER6_ID,TIMER7_ID,TIMER8_ID,TIMER9_ID,TIMER10_ID,TIMER11_ID] **u32_time:** delay time in seconds
Parameters (out):	None
Return	INVALID_OPERATION: in case invalid passing parameter VALID_OPERATION: in case valid passing parameter

Application flowchart



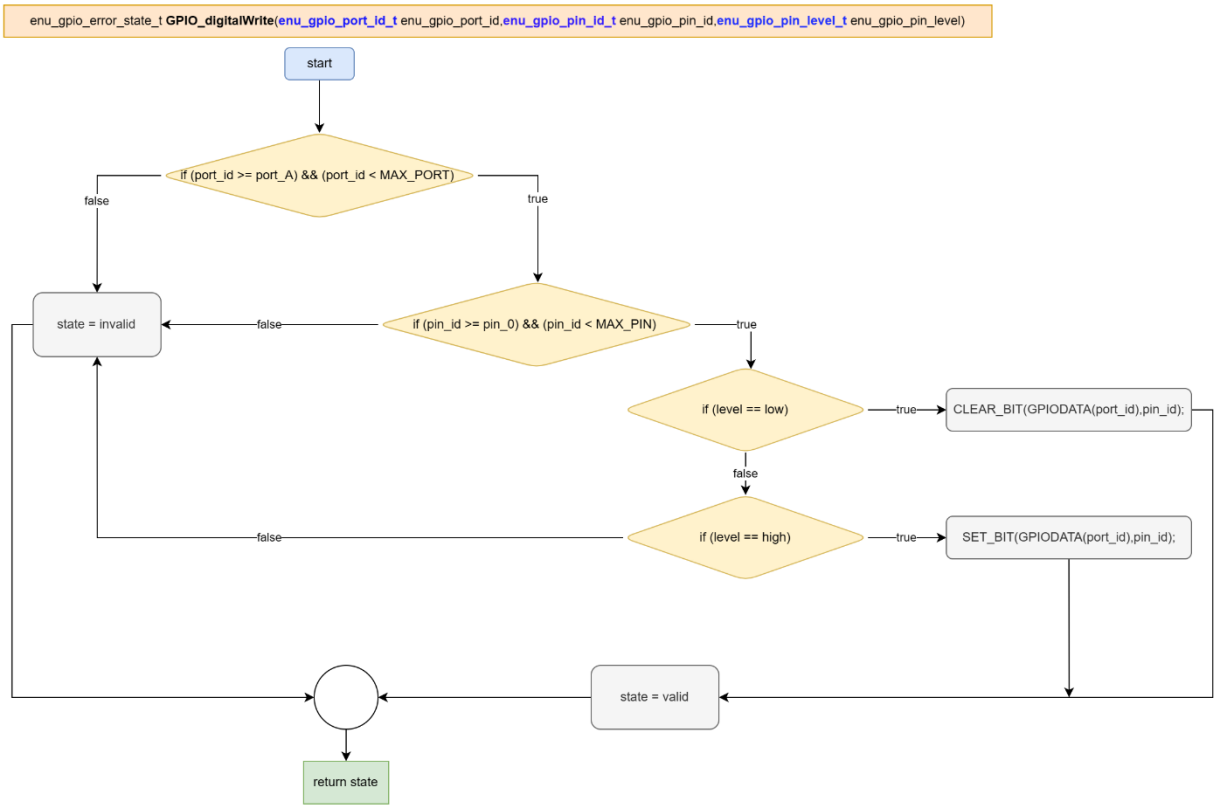
GPIO flowchart

- **GPIO_Init**

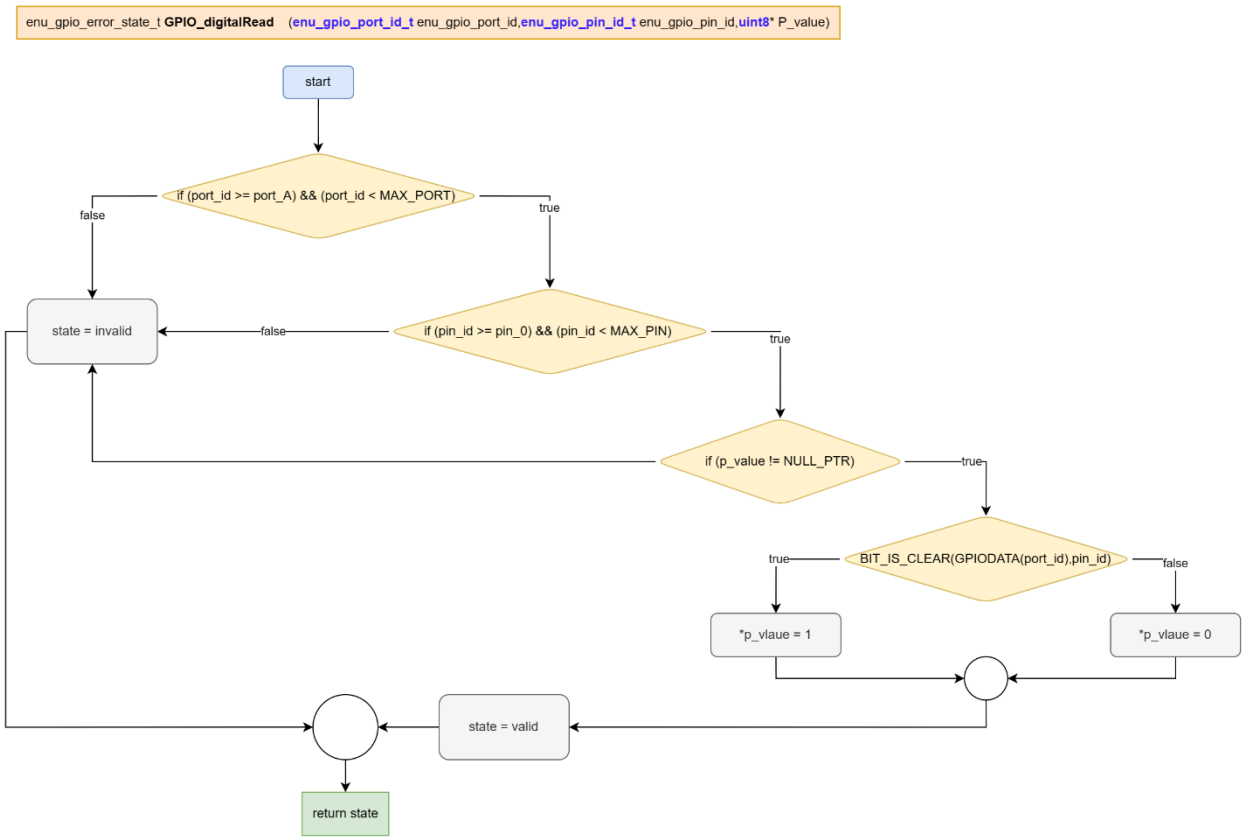


GPIO flowchart

- **GPIO_digitalWrite**

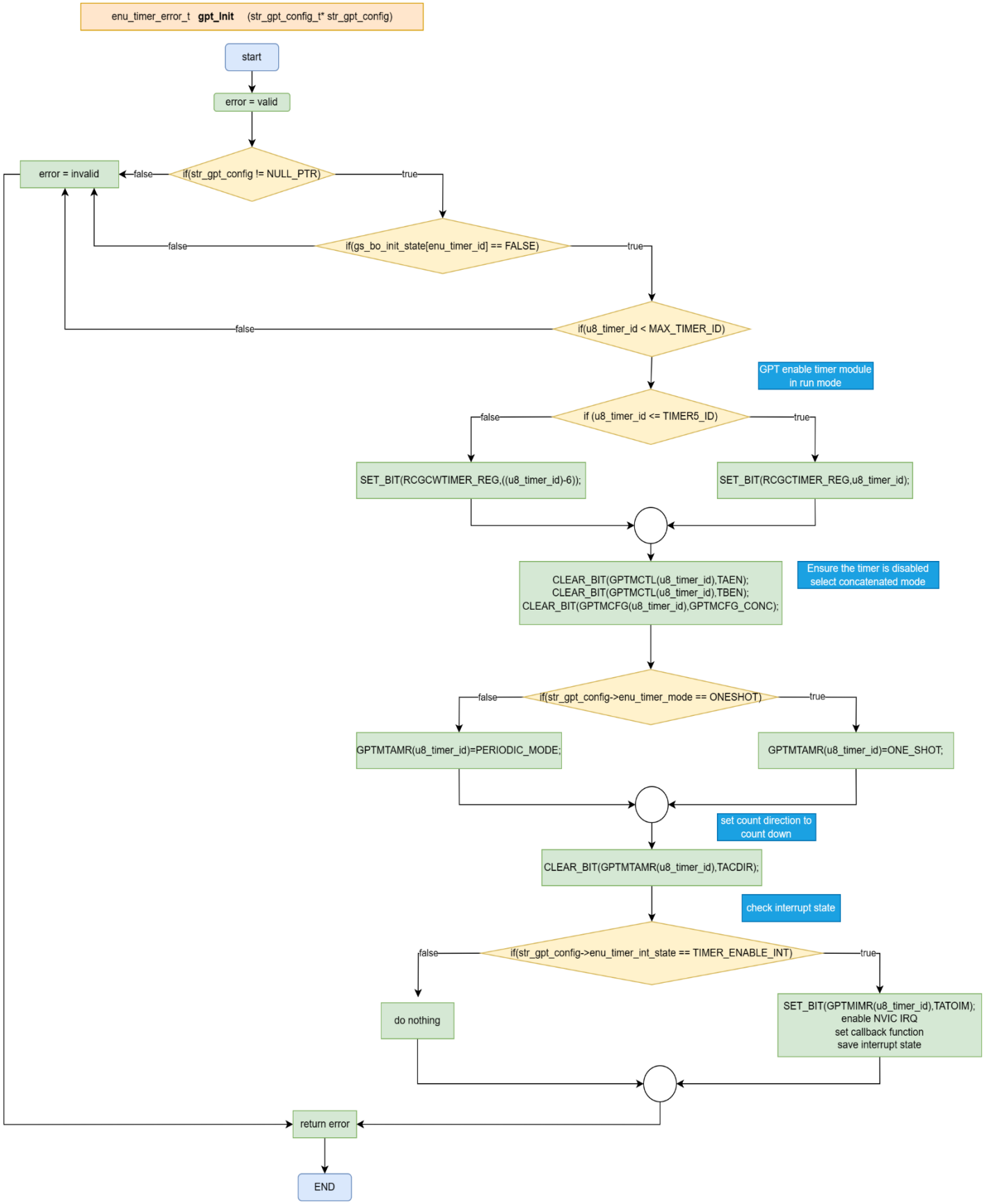


- **GPIO_digitalRead**



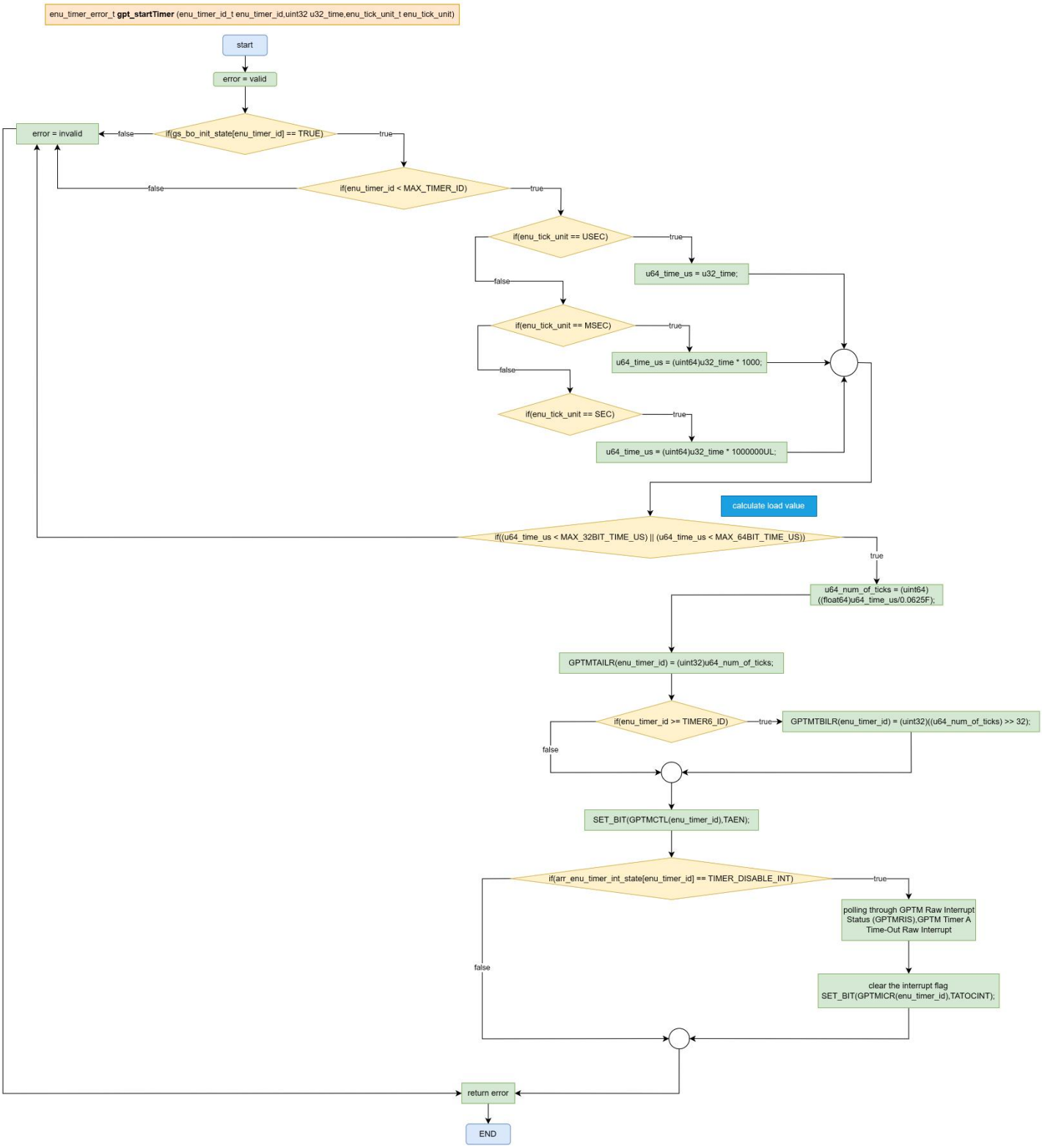
GPT flowchart

- **gpt_init**

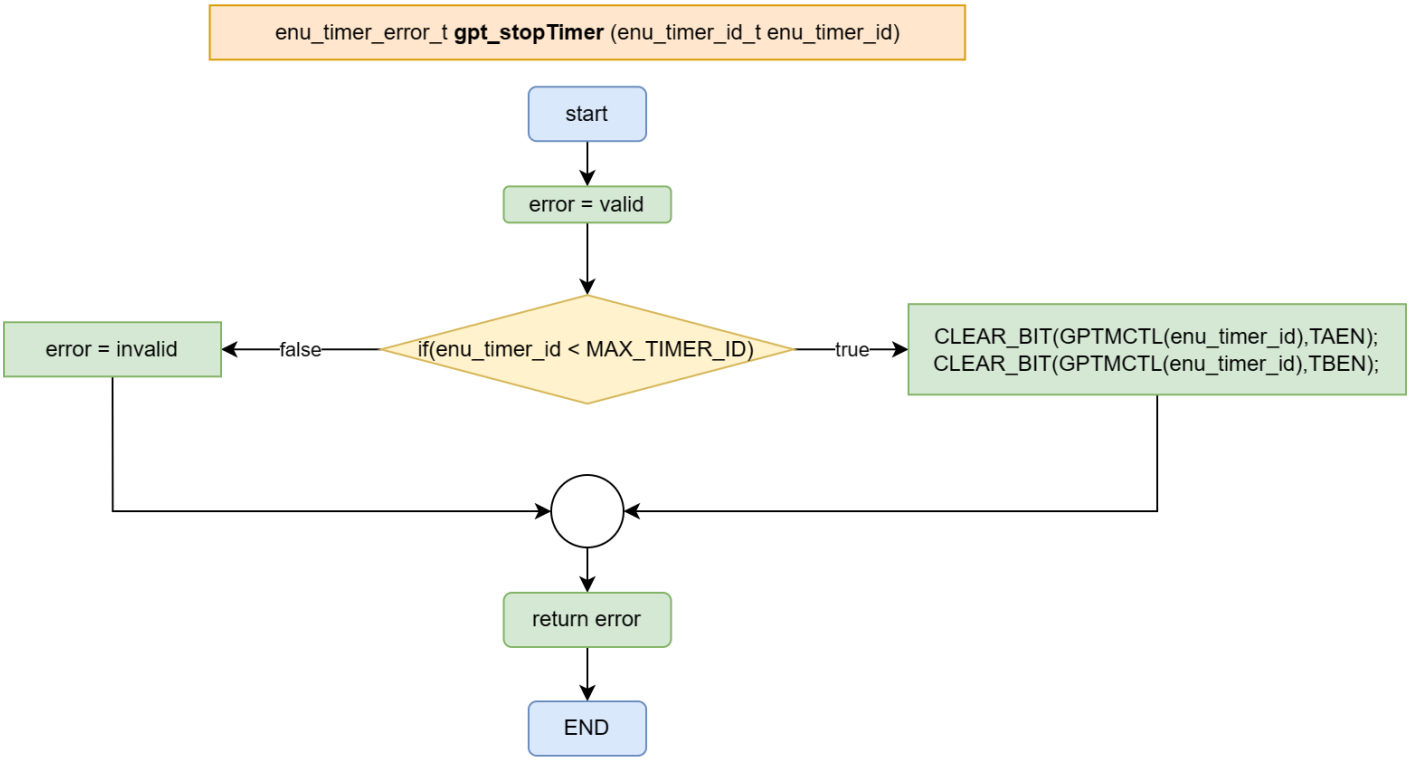


GPT flowchart

- **gpt_startTimer**

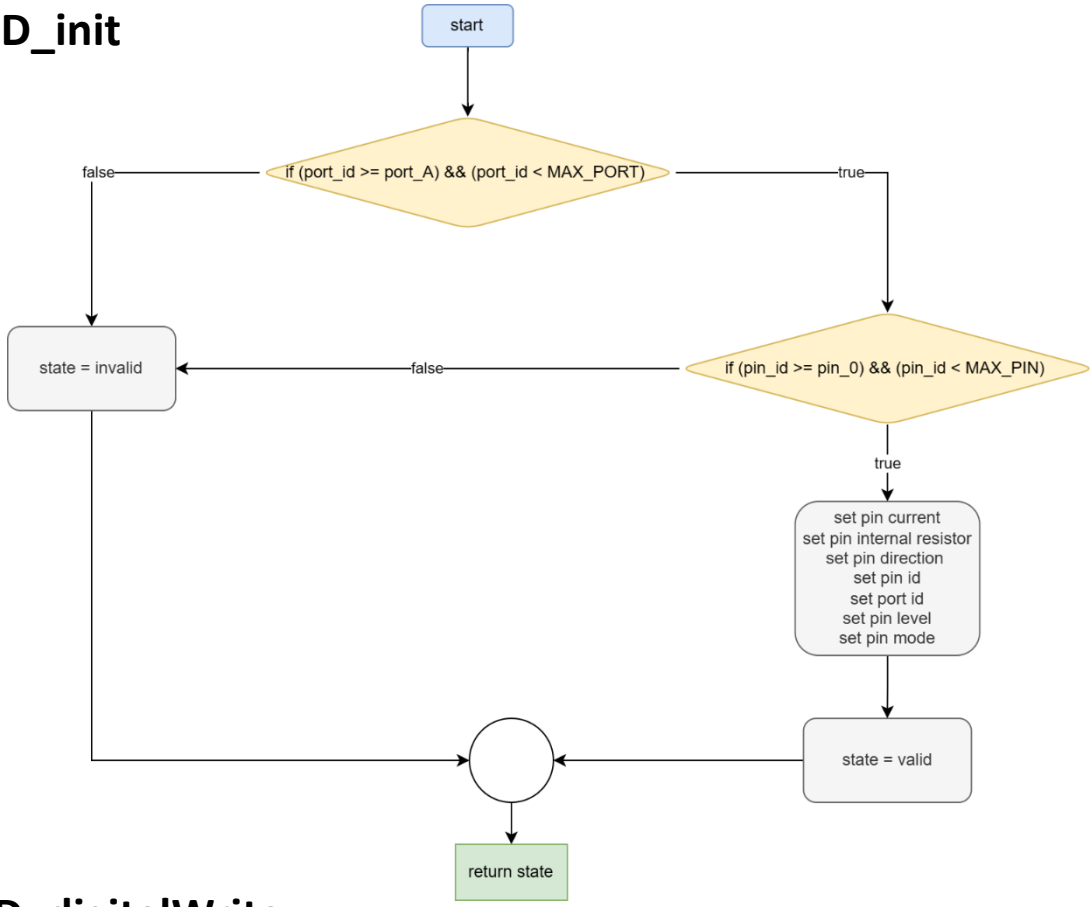


- **gpt_stopTimer**

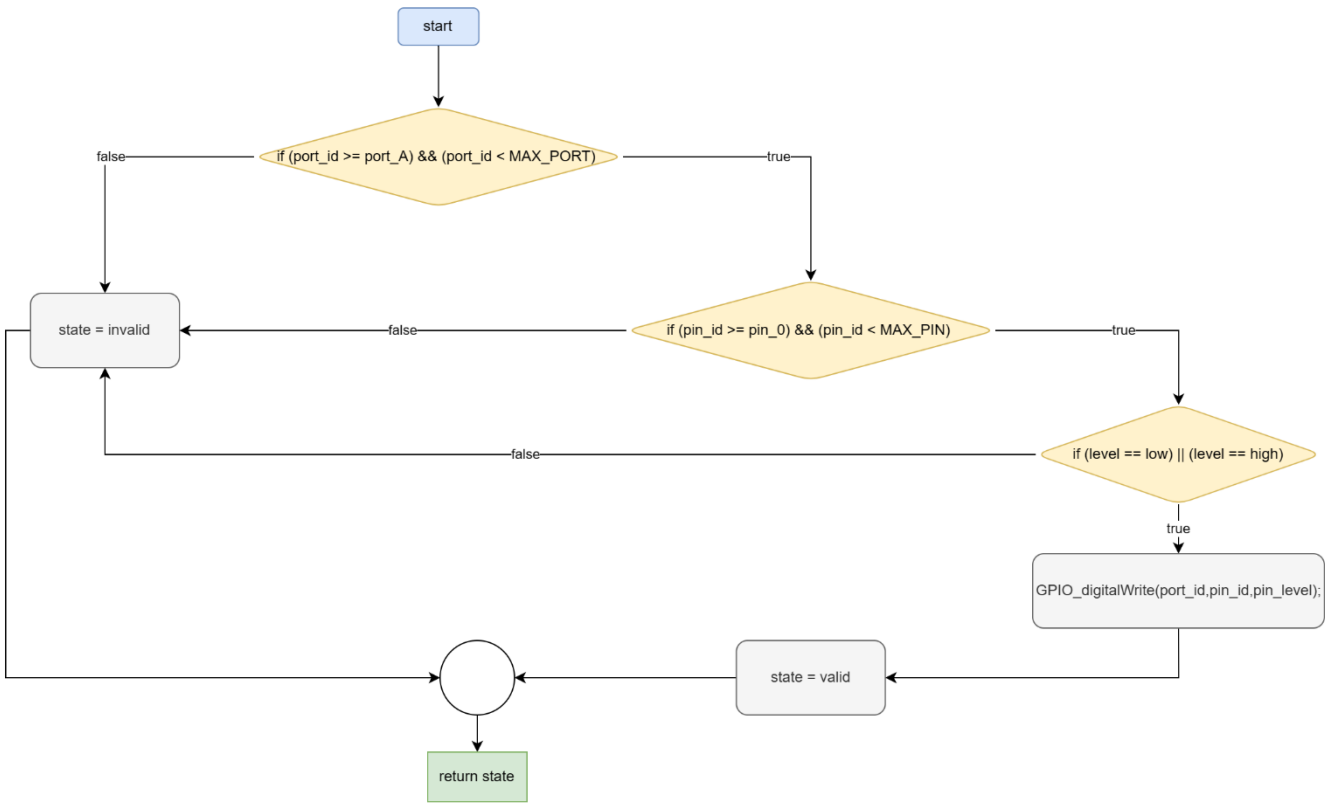


LED flowchart

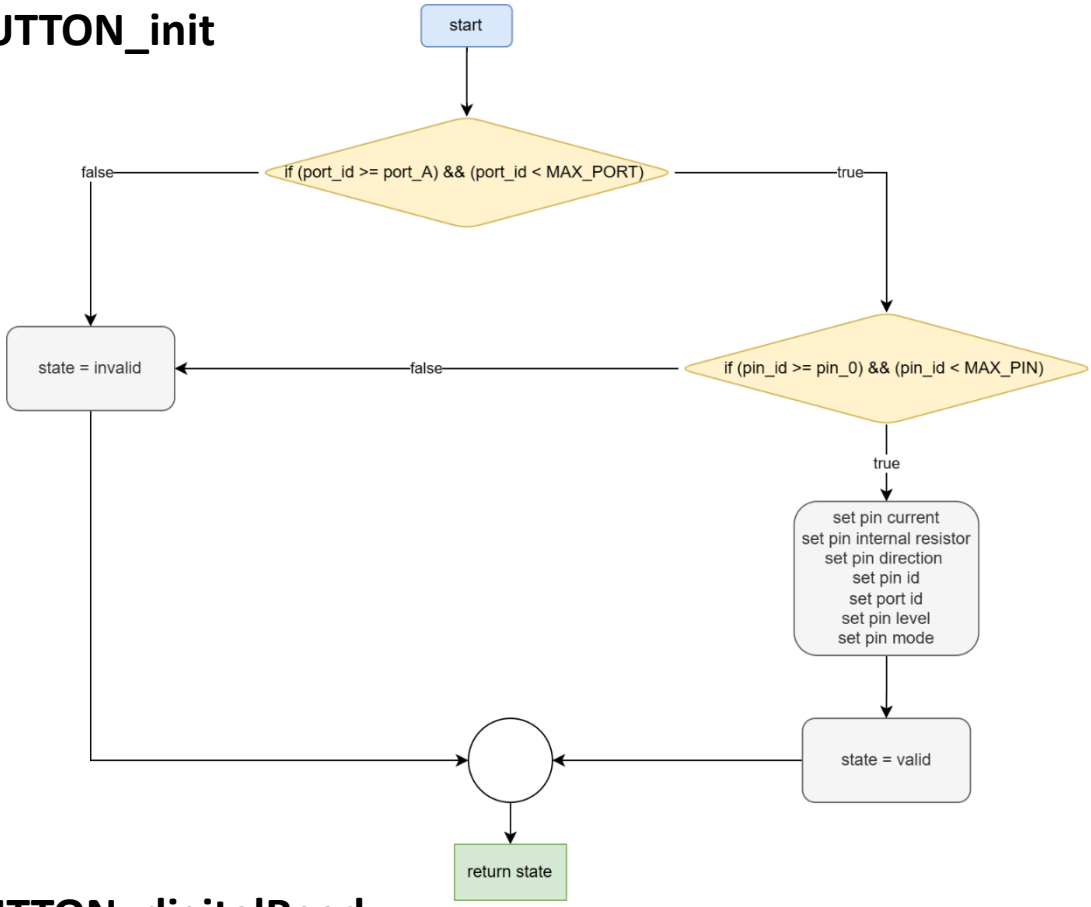
- **LED_init**



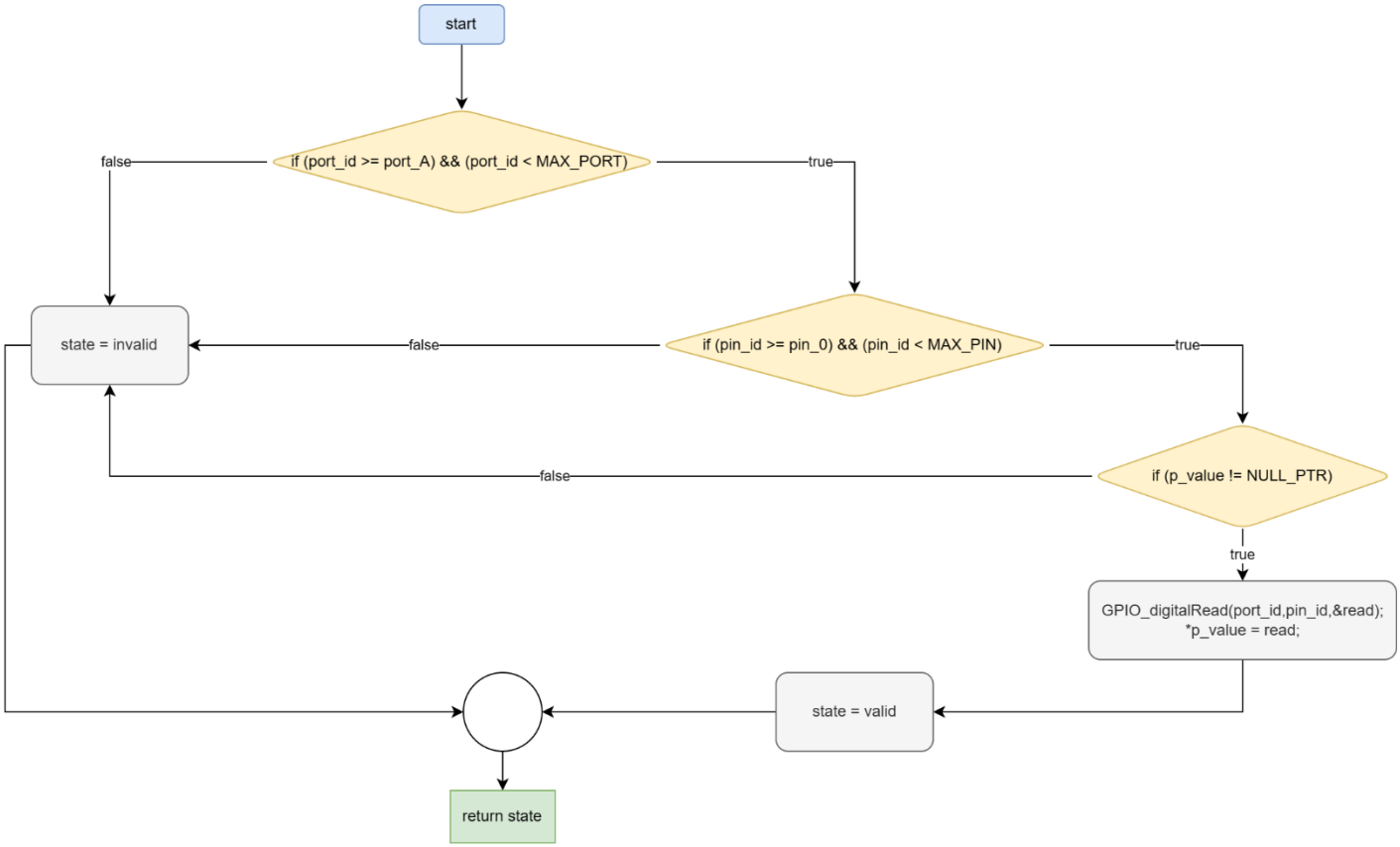
- **LED_digitalWrite**



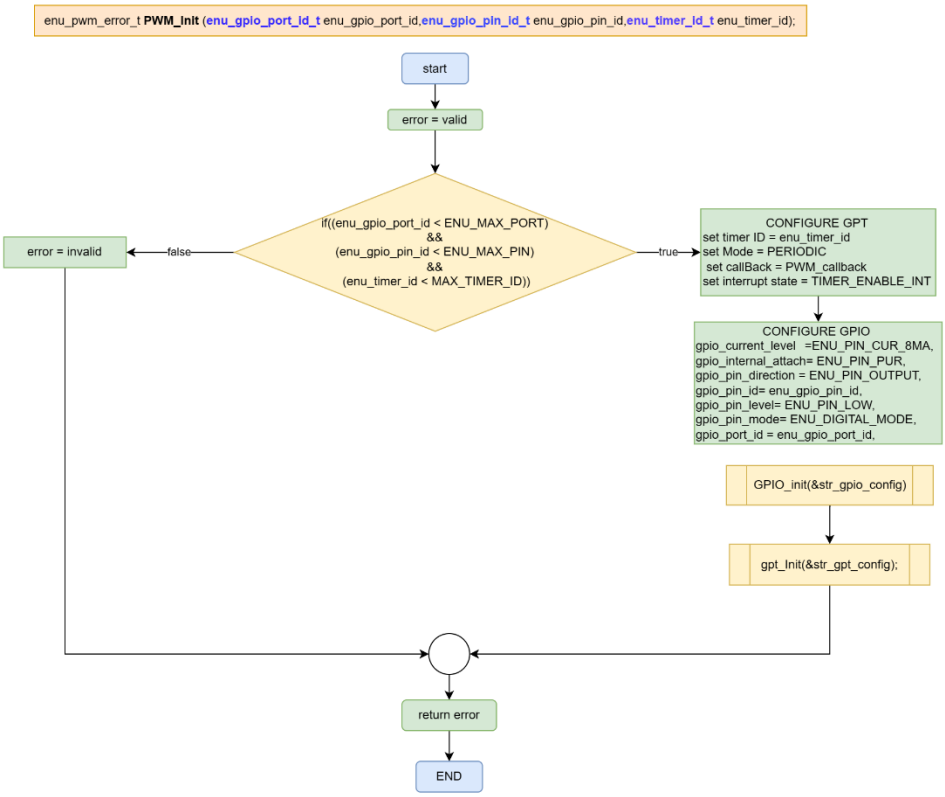
- **BUTTON_init**



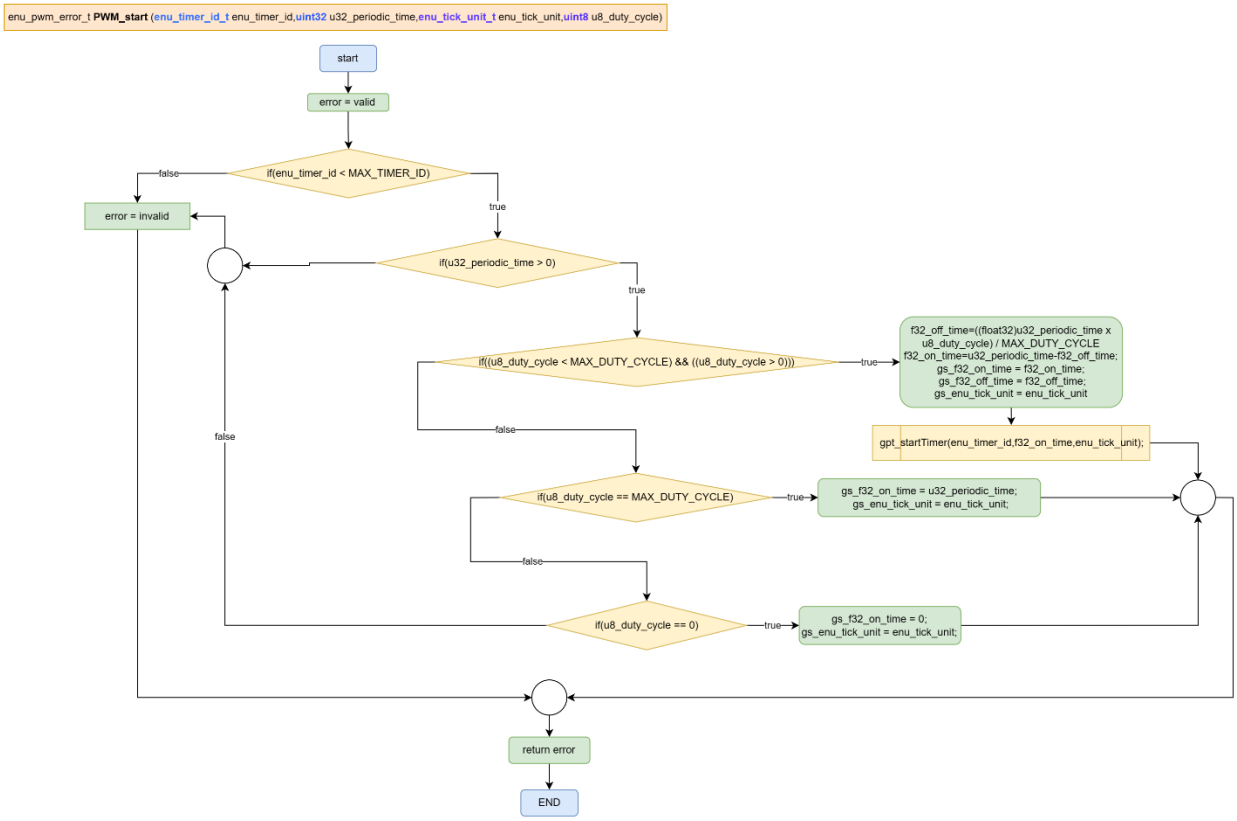
- **BUTTON_digitalRead**



- **PWM_init**

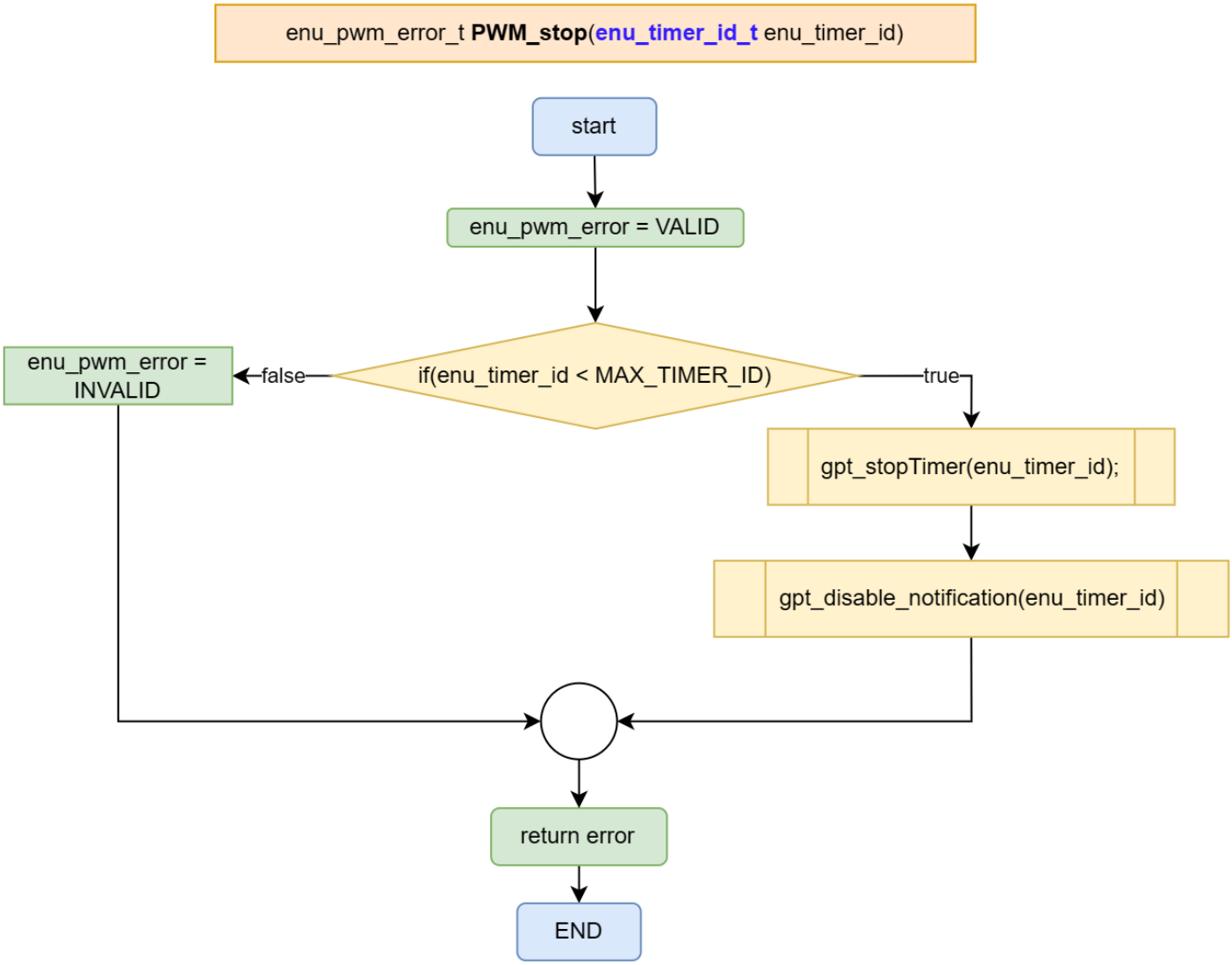


- **PWM_start**



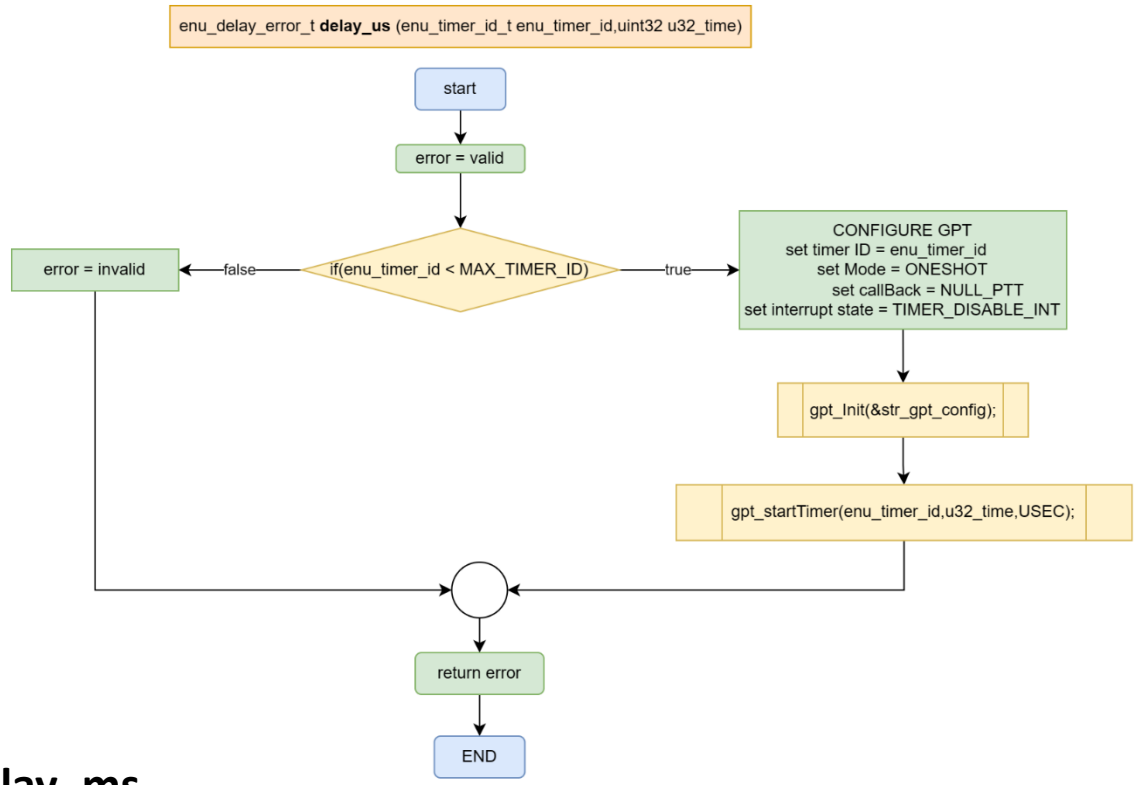
PWM flowchart

- **PWM_stop**

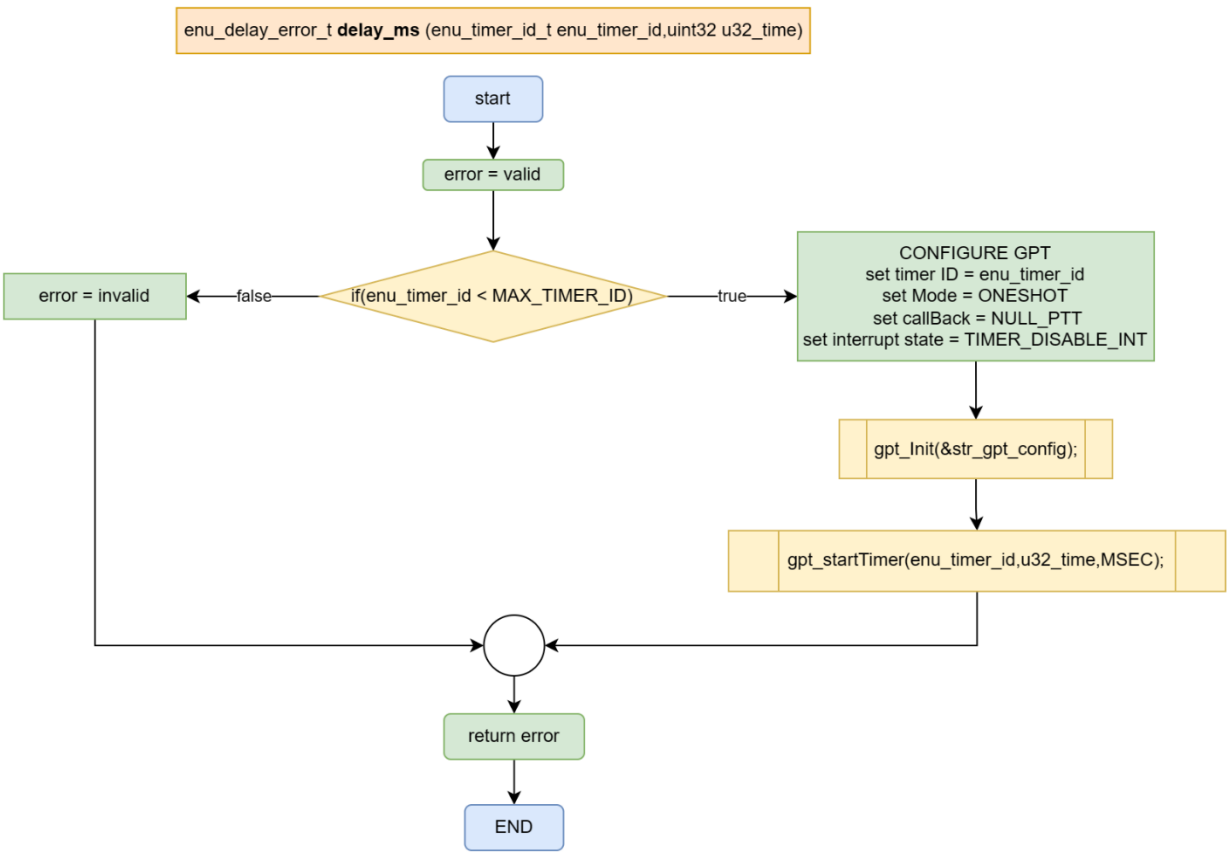


Delay flowchart

- **Delay_us**



- **Delay_ms**



Delay flowchart

- **Delay_sec**