LED CONTROL v1

AHMED HESHAM – SARAH MOHAMMED
AUTOMOTIVE EMBEDDED BOOTCAMP
6/18/23

LED CONTROL DESIGN v1

1. Project Introduction

The project to develop GPIO driver and use it to control RGB LED on the TivaC board based using the push button.

1.1. Project Components

Tiva C verification board using SW1 and rgb LED

1.2. Project Requirements

1. The RGB LED is OFF initially
2. Pressing SW1:

   2.1. After the first press, the Red led is on

   2.2. After the second press, the Green Led is on

   2.3. After the third press, the Blue led is on

   2.4. After the fourth press, all LEDs are on

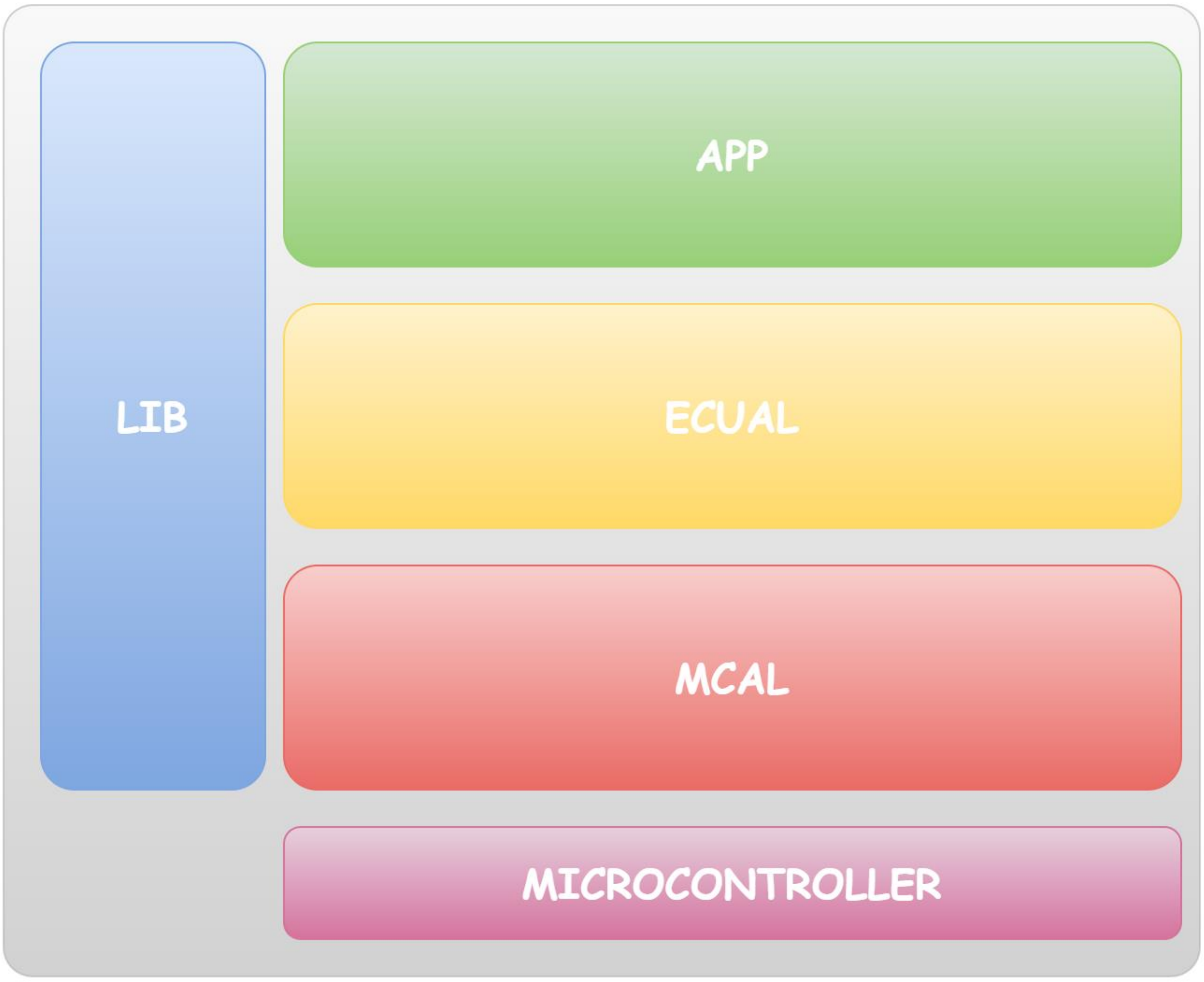
   2.5. After the fifth press, should disable all LEDs

   2.6. After the sixth press, repeat steps from 1 to 6

2. High Level Design

2.1. System Architecture

2.1.1. Layered Architecture



2.2. Modules Description

2.2.1. GPIO

The *GPIO* module reads input signals from the system's sensors (such as buttons) and drives output signals to the system's actuators (such as *LEDs*). It provides a set of APIs to configure the direction and mode of each pin (input/output, pull-up/down resistor), read the state of an input pin, and set the state of an output pin.

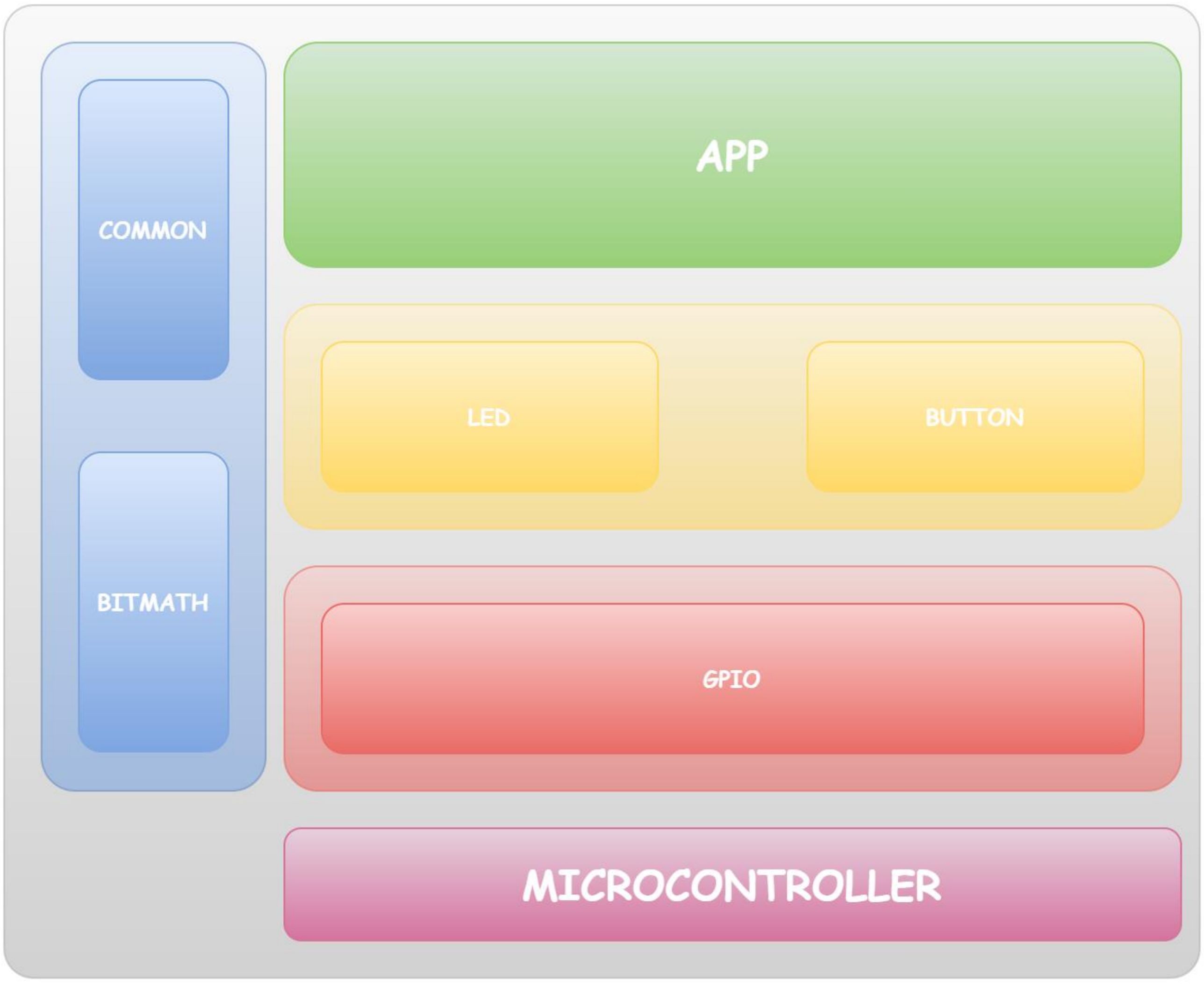
2.2.3. LED Module

The LED module is responsible for initializing the LEDs and turning them on/off or toggling them.

2.2.2. BTN Module

The *BTN* (Button) module is responsible for reading the state of the system's buttons. It provides a set of APIs to enable/disable button interrupts, set the button trigger edge (rising/falling/both), and define an ISR that will be executed when a button press is detected.

2.2.4. Design



2.3. Drivers’ Documentation (APIs)

2.3.1 Definition

An *API* is an *Application Programming Interface* that defines a set of *routines*, *protocols* and *tools* for creating an application. An *API* defines the high level interface of the behavior and capabilities of the component and its inputs and outputs.

An *API* should be created so that it is generic and implementation independent. This allows for the API to be used in multiple applications with changes only to the implementation of the API and not the general interface or behavior.

2.3.2. MCAL APIs

2.3.2.1. GPIO Driver

```
| Brief
|              This function is used to initialize a specific pin
| Parameters
|           [in]      ptr_str_gpio_config: Ptr to the gpio instance structure
|           [out]     none
| Return
|             ERROR_OK : In case of successeion
|             GPIO_INVALID_PIN_INDEX : In case of wrong pin index
|             GPIO_INVALID_PIN_MODE : In case of wrong mode choosen
|             GPIO_INVALID_PIN_DIRECTION : In case of wrong direction choosen
|             GPIO_INVALID_OP_CURRENT      :In case of wrong current choosen
|             GPIO_INVALID_INTERNAL_ATTACH : In case of wrong internal attach
|
enu_error_status_t_ gpio_pin_init(str_gpio_config_t_* ptr_str_gpio_config)

| Brief
|                    This function is used to Write a specific output on a pin
| Parameters
|           [in]            uint8_pin_index : Pin index used
|                    enu_pin_level : Output level to be written
|            [out]   none
| Return
|                    ERROR_OK:In case of successeion
|                    GPIO_INVALID_PIN_INDEX:In case of wrong pin index
|                    GPIO_INVALID_PIN_LEVEL:In case of wrong output level choosen
|
enu_error_status_t_ gpio_pin_write(uint8_t_ uint8_pin_index, enu_gpio_pin_level_t_ enu_pin_level)




| Brief
|                     This function is used to toggle output level on a pin
| Parameters
|           [in]              uint8_pin_index:Pin Index used
|            [out]    none
|
| Return
|                     ERROR_OK:In case of successeion
|                     GPIO_INVALID_PIN_INDEX:In case of wrong pin index
|
enu_error_status_t_ gpio_pin_toggle(uint8_t_ uint8_pin_index)

| Brief
|                     This function is used to get pin's input value
| Parameters
|           [in]      uint8_pin_index:Pin index used
|            [out]    uint8_pin_state:Pin level state
| Return
|                     ERROR_OK:In case of successeion
|                     GPIO_INVALID_PIN_INDEX:In case of wrong pin index
|
enu_error_status_t_ gpio_pin_read(uint8_t_ uint8_pin_index, uint8_t_* uint8_pin_state)

| Brief
|                     This function is used to enable pin's interrupt
| Parameters
|           [in]      uint8_pin_index:Pin index used
|            [out]    none
| Return
|                     ERROR_OK:In case of successeion
|                     GPIO_INVALID_PIN_INDEX:In case of wrong pin index
|
enu_error_status_t_ gpio_pin_enable_notification(uint8_t_ uint8_pin_index)

| Brief
|      This function is used to set callback function for a specific pin's
|     interrupt handler
| Parameters
|           [in]              uint8_pin_index:Pin index used
|                     ptr_callback:pointer to a callback function
|            [out]    none
| Return
|                     ERROR_OK:In case of successeion
|                     GPIO_INVALID_PIN_INDEX:In case of wrong pin index
|                     PASSING_NULL_PTR:In case of passing null pointer
|
enu_error_status_t_ gpio_pin_set_callback(uint8_t_ uint8_pin_index, ptr_gpio_callback_t_
ptr_callback)
```

2.3.3. HAL APIs

2.3.3.1. LED APIs

```
 |     @brief              This function is initalize pin
 |     @param [in]         none
 |     @return                    ERROR_OK      :      In case of succeseion
 |     @return                    LED_NOK     : In case of wrong pin index

enu_error_status_t_ led_init(void);


 |     @brief              This function is turn on led
 |     @param [in]         uint8_ledpin_index         :      Pin index used
 |     @return                    ERROR_OK       :     In case of succeseion
 |     @return                    LED_NOK        :     In case of wrong pin index

enu_error_status_t_ led_on(uint8_t_ uint8_ledpin_index );


 |     @brief              This function is turn off led
 |     @param [in]         uint8_ledpin_index: Pin index used
 |     @return                    ERROR_OK       :In case of succeseion
 |     @return                    LED_NOK      :      In case of wrong pin index

enu_error_status_t_ led_off(uint8_t_ uint8_ledpin_index );
```

2.3.3.2. BTN APIs

```
 |     @brief              This function is used for initalizing button
 |     @param [in]         ptr_func               :   pointer to callback
 |     @return                    ERROR_OK       :    In case of succeseion
 |  @return                 PASSING_NULL_PTR       :   In case of passing null pointer
 |     @return                    BUTTON_NOK     :    In case of wrong pin index

enu_error_status_t_ button_init(void(*ptr_func)(void));


 |     @brief              This function is used for getting button  state
 |     @param [in]         buttonpin                          :  Pin index used
 |     @param [out] ptr_uint8_button_state         :  returns button state
 |     @return                    ERROR_OK            :In case of succeseion
 |     @return                    BUTTON_NOK    :In case of wrong pin index

enu_error_status_t_ button_get_state(uint8_t_ uint8_button_pin, uint8_t_* ptr_uint8_button_state);
```

2.3.4. APP APIs

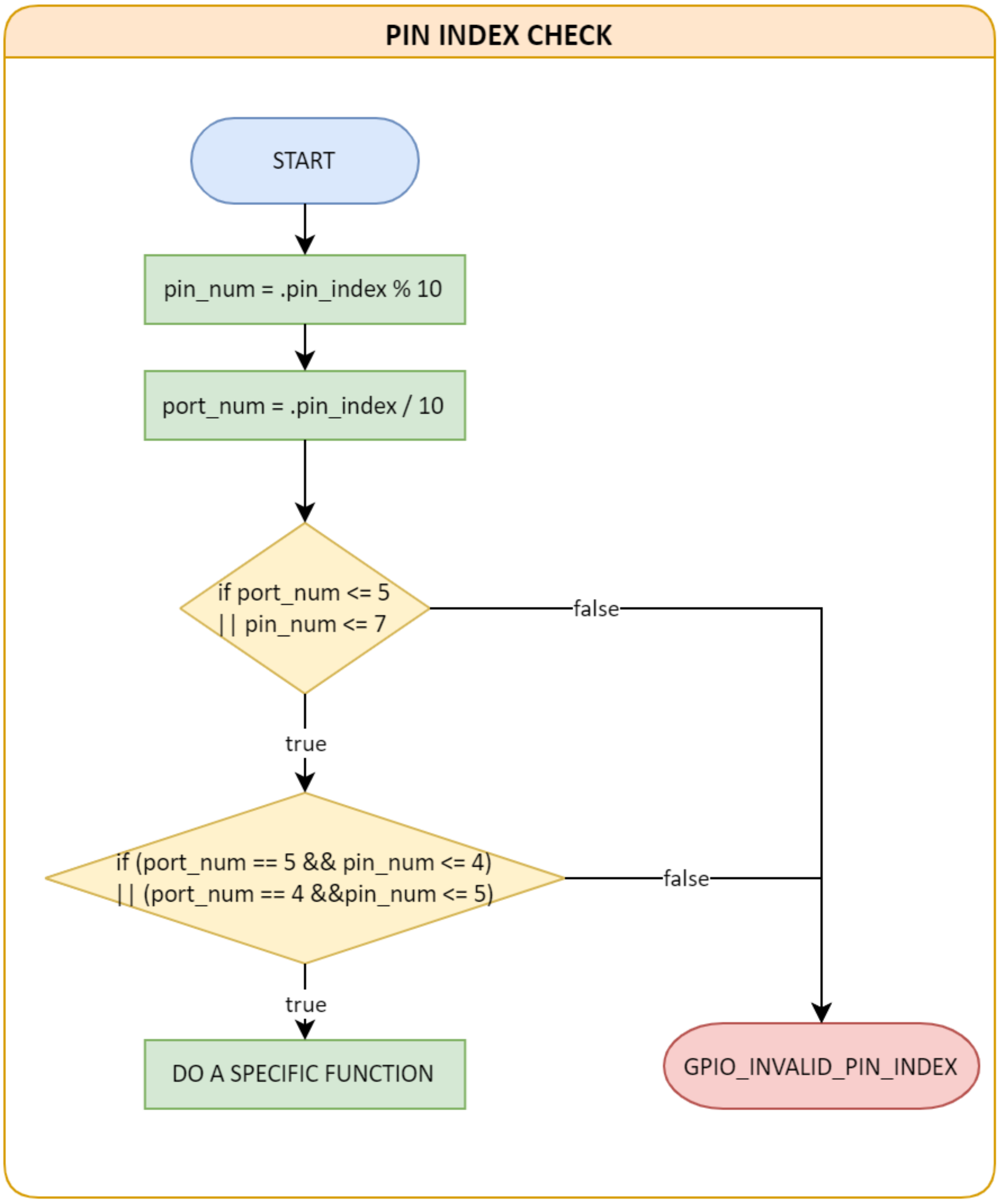
```
| Brief
|                   This function is used to initialize drivers used
| Parameters
|      [in]         none
|      [out]   none
|
| Return
|                   none
|
void app_init(void)

| Brief
|                   This function is the called back when button pressed
| Parameters
|      [in]         none
|      [out]   none
|
| Return
|                   none
|
static void button_task(void)
```

3. Low Level Design

3.1. MCAL Layer

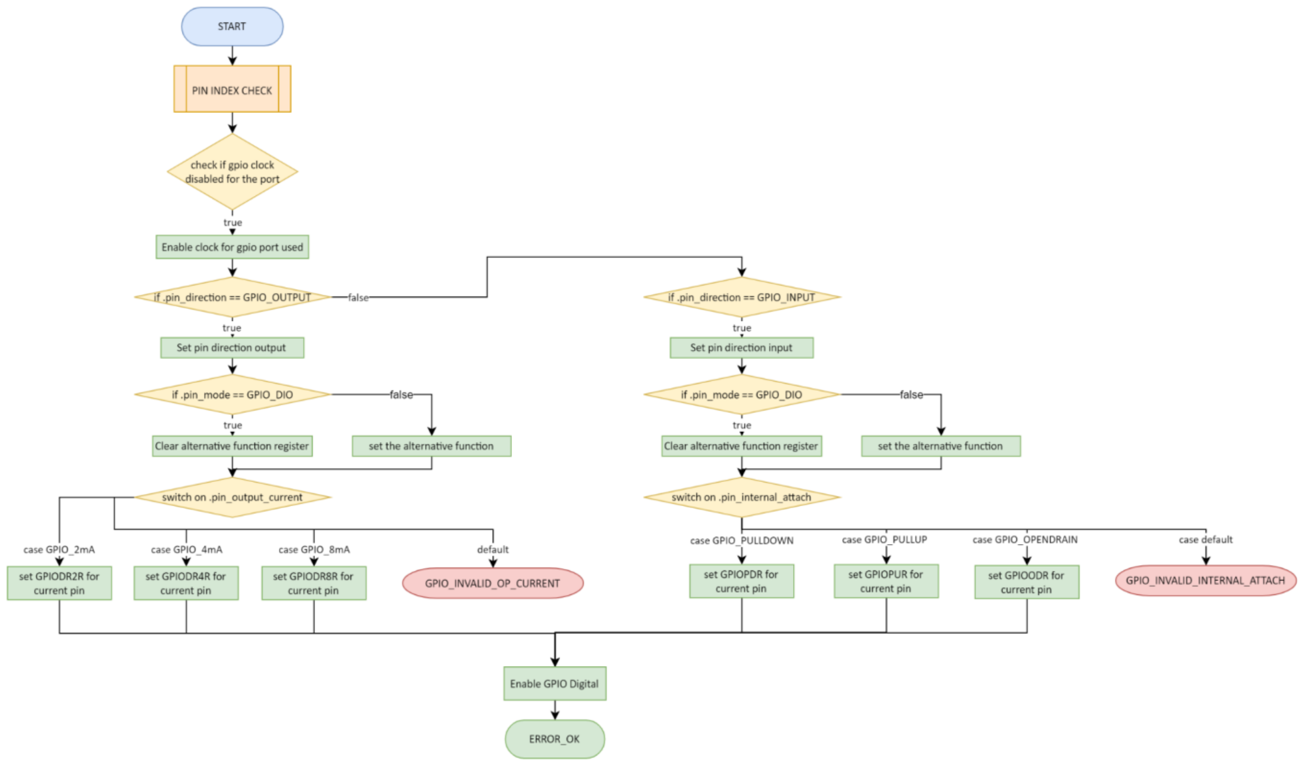
3.1.1. GPIO Module

3.1.1.a. PIN INDEX CHECK sub-process

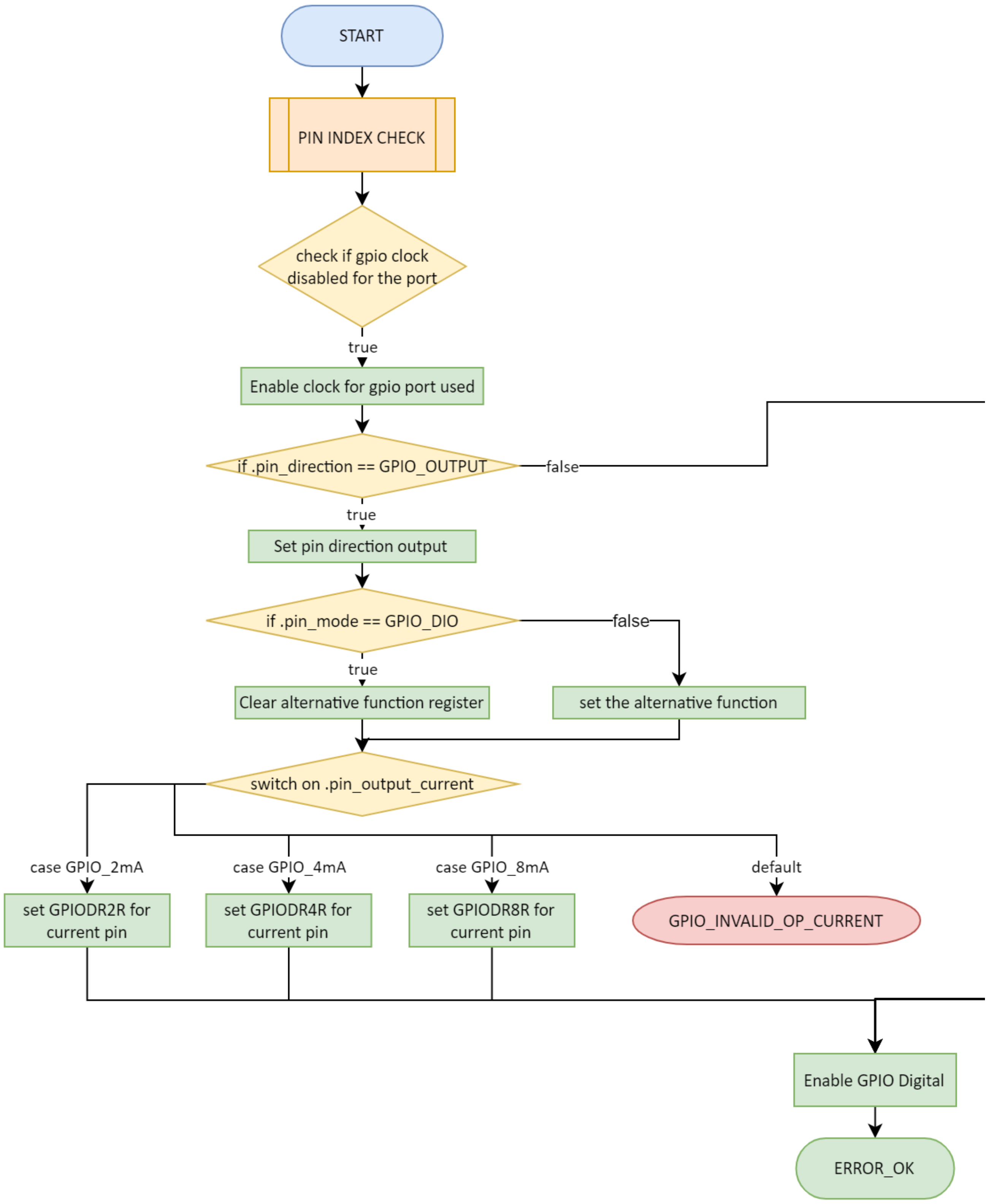

PIN INDEX CHECK
START
pin_num = .pin_index % 10
port_num = .pin_index / 10
if port_num <= 5 || pin_num <= 7
false
true
if (port_num == 5 && pin_num <= 4) || (port_num == 4 &&pin_num <= 5)
false
true
DO A SPECIFIC FUNCTION
GPIO_INVALID_PIN_INDEX

3.1.1.1. gpio_pin_init

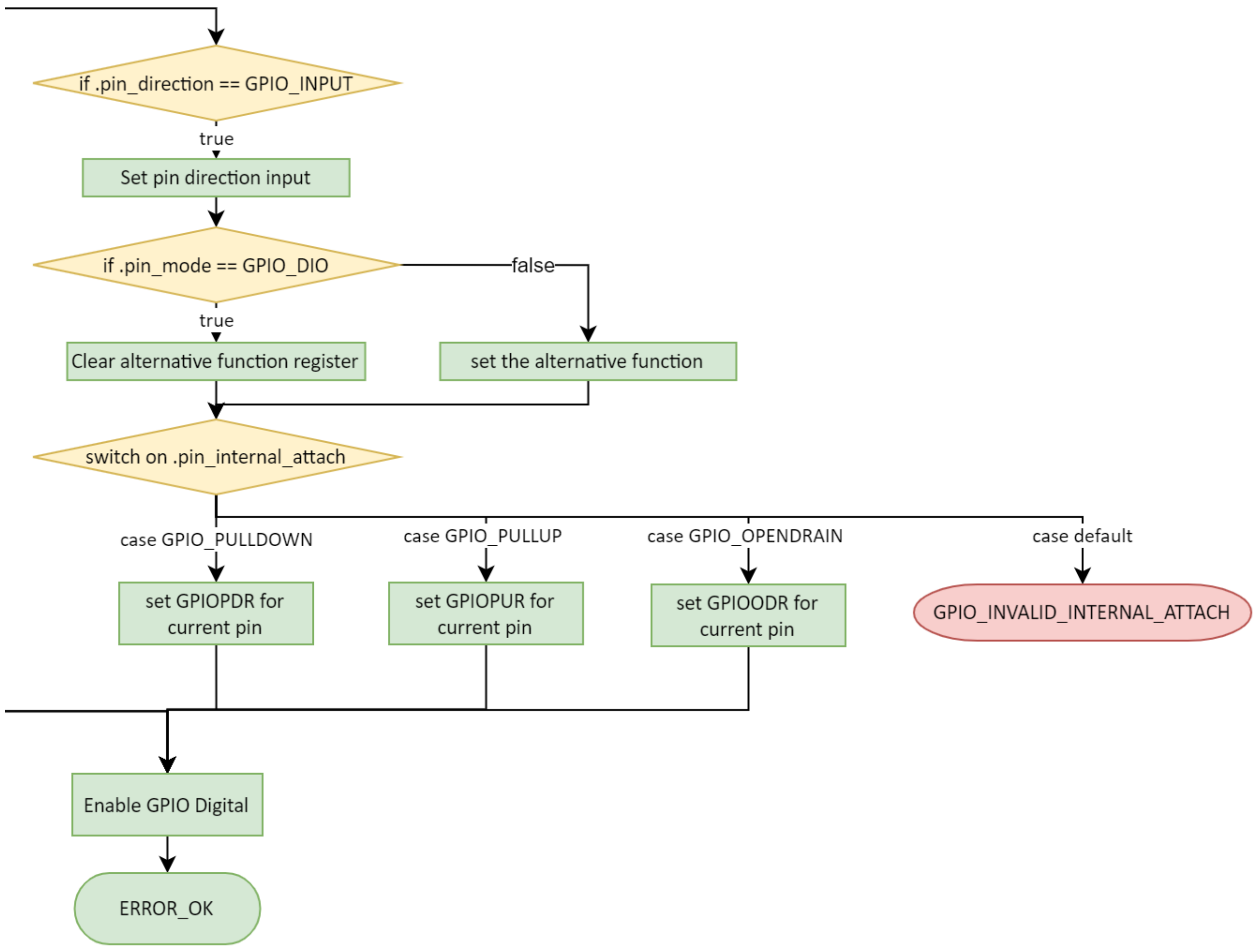
Full diagram
Part 1 and part 2 in the next pages


START
PIN INDEX CHECK
check if gpio clock disabled for the port
true
Enable clock for gpio port used
if .pin_direction == GPIO_OUTPUT
false
true
Set pin direction output
if .pin_mode == GPIO_DIO
false
true
Clear alternative function register
set the alternative function
switch on .pin_output_current
case GPIO_2mA
case GPIO_4mA
case GPIO_8mA
default
set GPIODR2R for current pin
set GPIODR4R for current pin
set GPIODR8R for current pin
GPIO_INVALID_OP_CURRENT
if .pin_direction == GPIO_INPUT
true
Set pin direction input
if .pin_mode == GPIO_DIO
false
true
Clear alternative function register
set the alternative function
switch on .pin_internal_attach
case GPIO_PULLDOWN
case GPIO_PULLUP
case GPIO_OPENDRAIN
case default
set GPIOPDR for current pin
set GPIOPUR for current pin
set GPIOODR for current pin
GPIO_INVALID_INTERNAL_ATTACH
Enable GPIO Digital
ERROR_OK

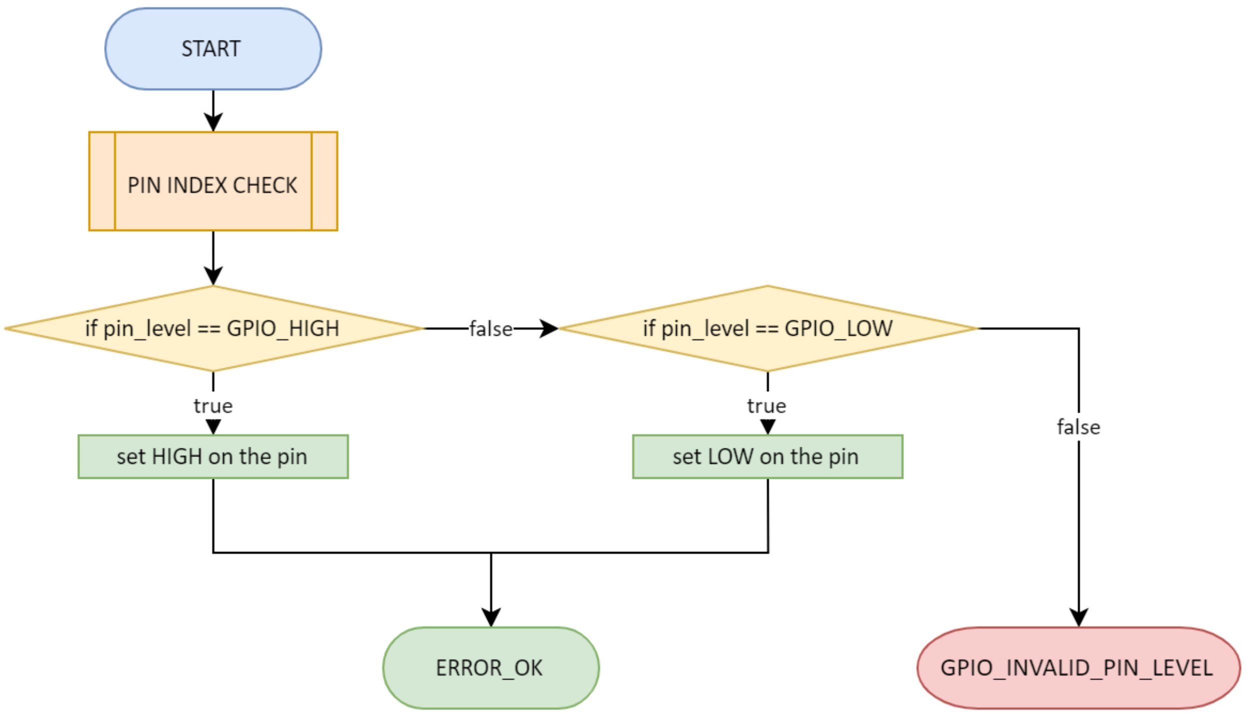
Part 1


START
PIN INDEX CHECK
check if gpio clock disabled for the port
true
Enable clock for gpio port used
if .pin_direction == GPIO_OUTPUT
false
true
Set pin direction output
if .pin_mode == GPIO_DIO
false
true
Clear alternative function register
set the alternative function
switch on .pin_output_current
case GPIO_2mA
case GPIO_4mA
case GPIO_8mA
default
set GPIODR2R for current pin
set GPIODR4R for current pin
set GPIODR8R for current pin
GPIO_INVALID_OP_CURRENT
Enable GPIO Digital
ERROR_OK

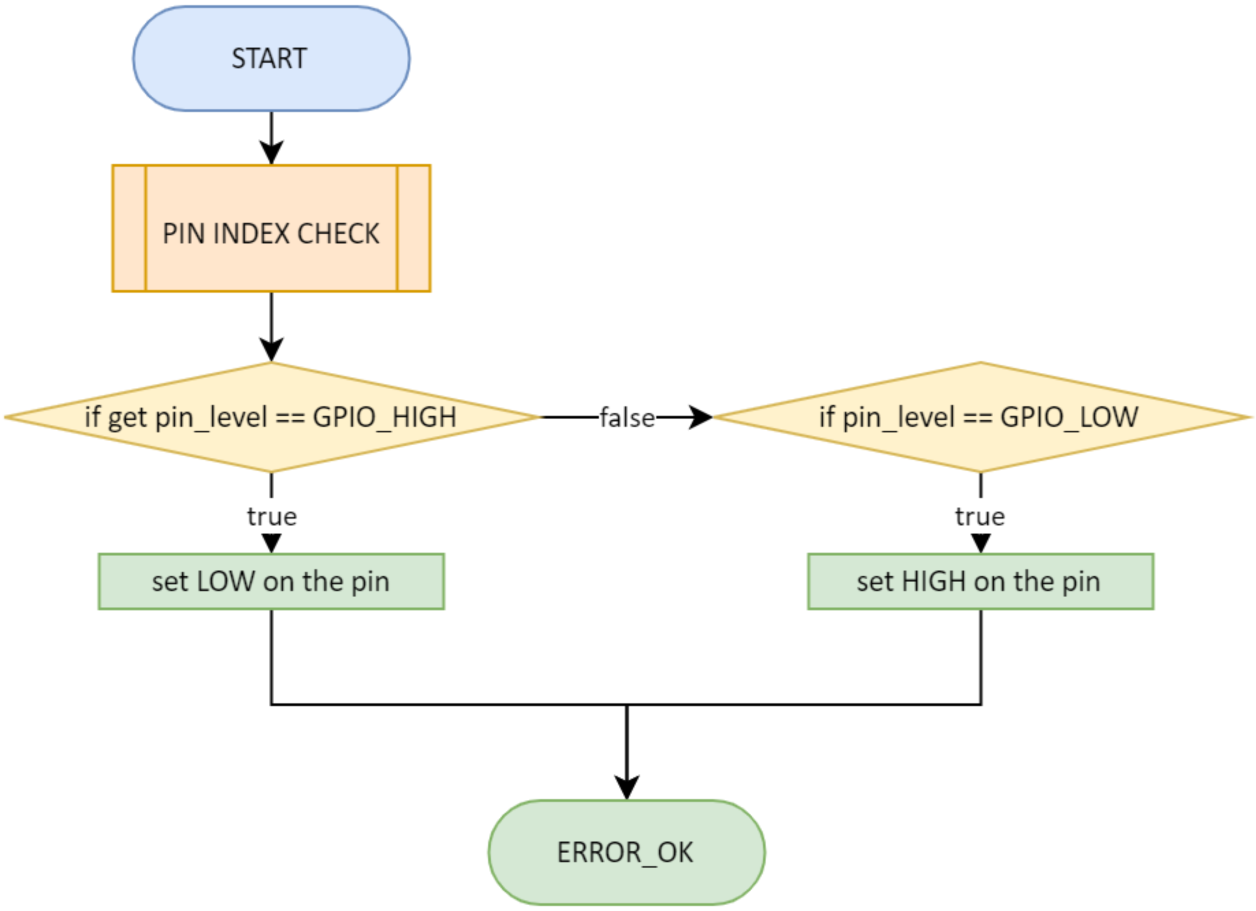
Part 2


if .pin_direction == GPIO_INPUT
true
Set pin direction input
if .pin_mode == GPIO_DIO
false
true
Clear alternative function register
set the alternative function
switch on .pin_internal_attach
case GPIO_PULLDOWN
case GPIO_PULLUP
case GPIO_OPENDRAIN
case default
set GPIOPDR for current pin
set GPIOPUR for current pin
set GPIOODR for current pin
GPIO_INVALID_INTERNAL_ATTACH
Enable GPIO Digital
ERROR_OK

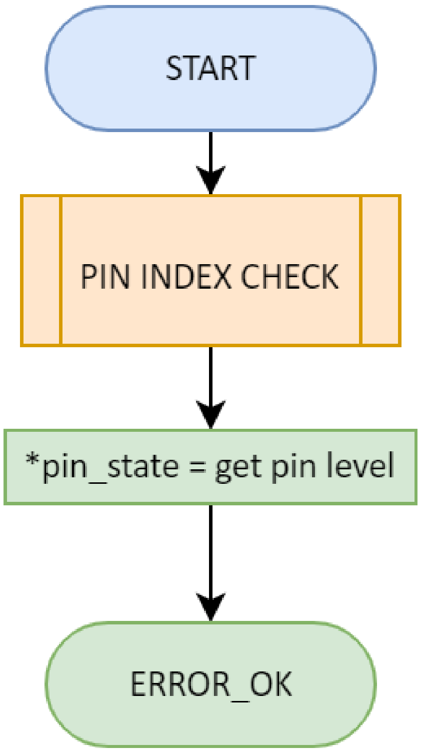
3.1.1.2. gpio_pin_write



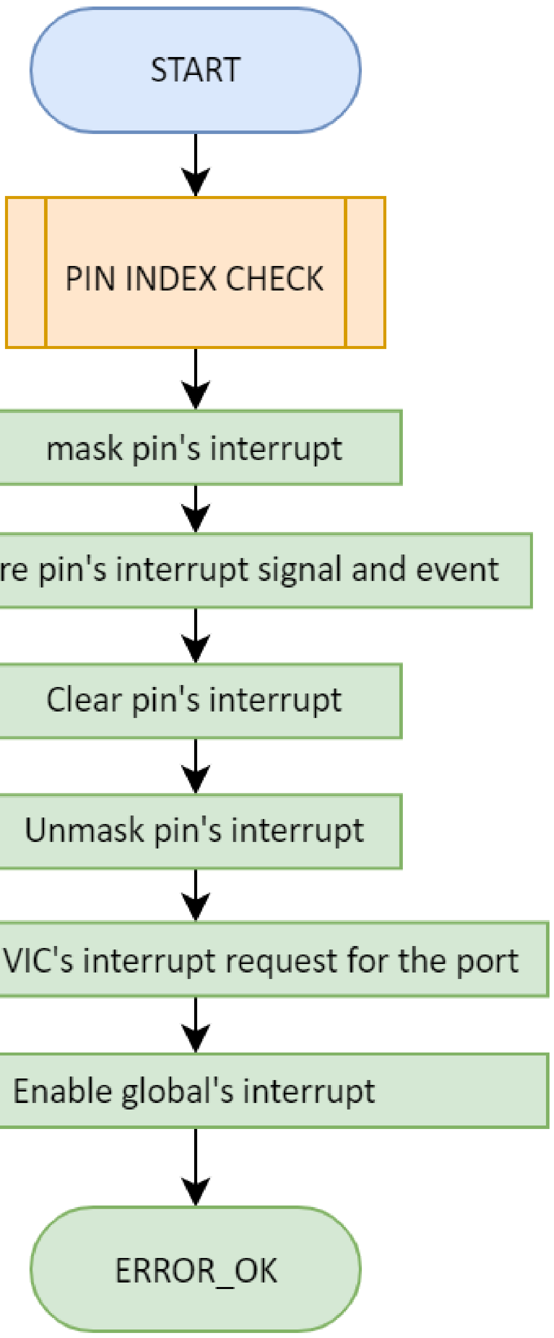
3.1.1.3. gpio_pin_toggle



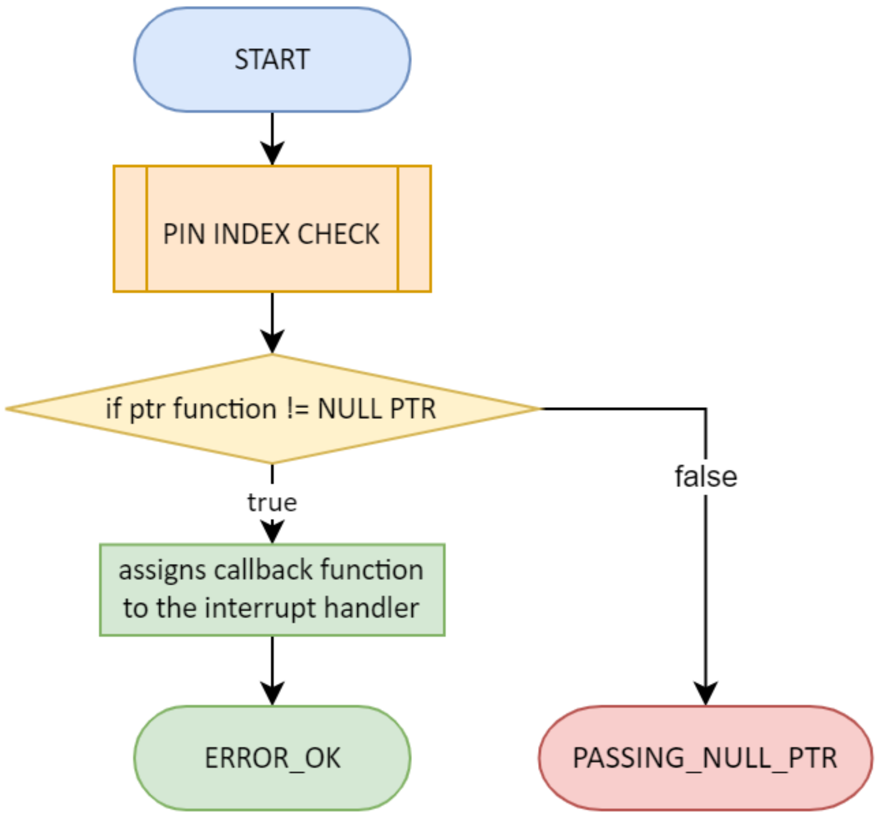
3.1.1.4. gpio_pin_read



3.1.1.5. gpio_pin_enable_notification


START
PIN INDEX CHECK
mask pin's interrupt
configure pin's interrupt signal and event
Clear pin's interrupt
Unmask pin's interrupt
enable NVIC's interrupt request for the port
Enable global's interrupt
ERROR_OK

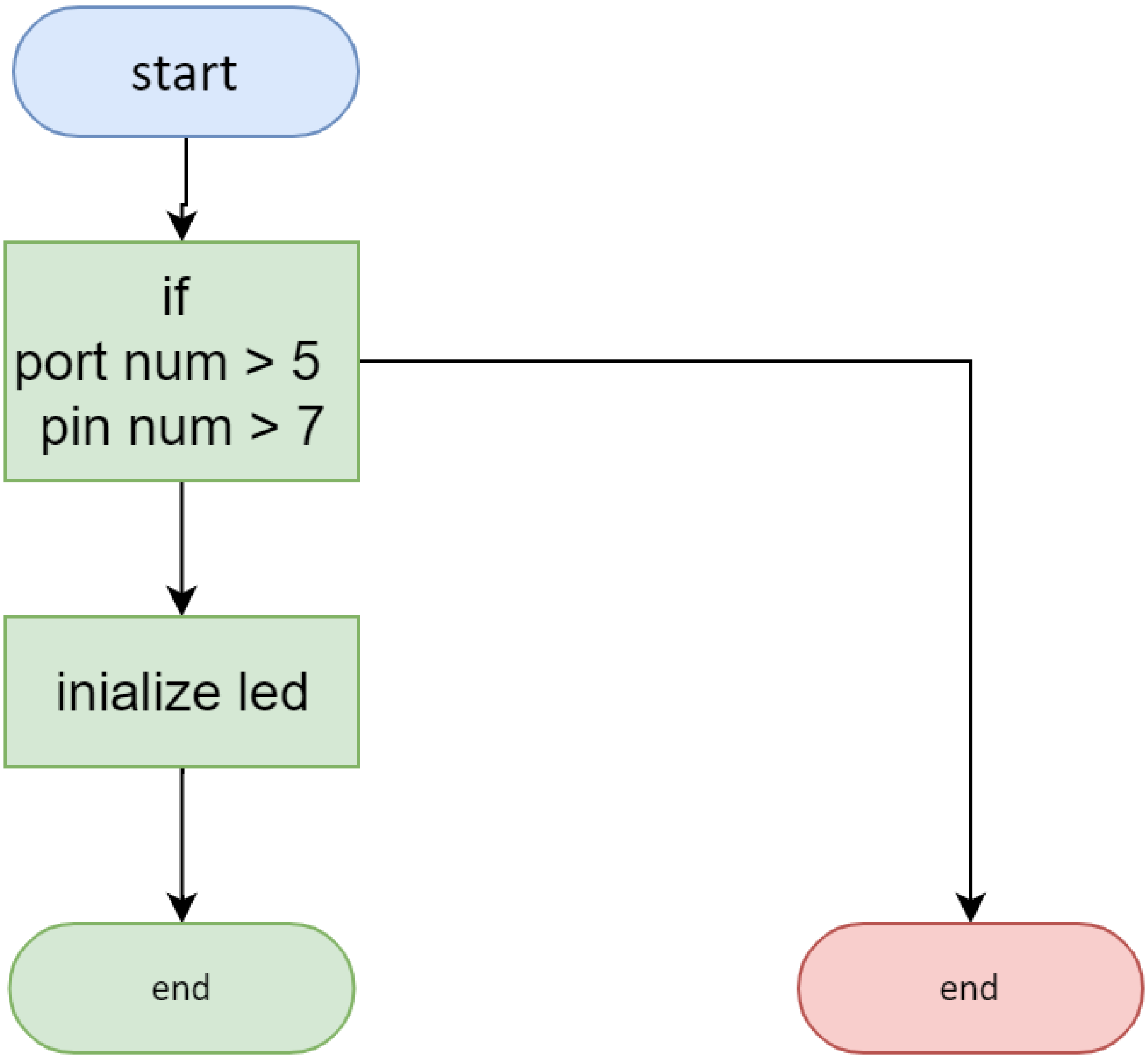
3.1.1.6. gpio_pin_set_callback


START
PIN INDEX CHECK
if ptr function != NULL PTR
true
false
assigns callback function to the interrupt handler
ERROR_OK
PASSING_NULL_PTR

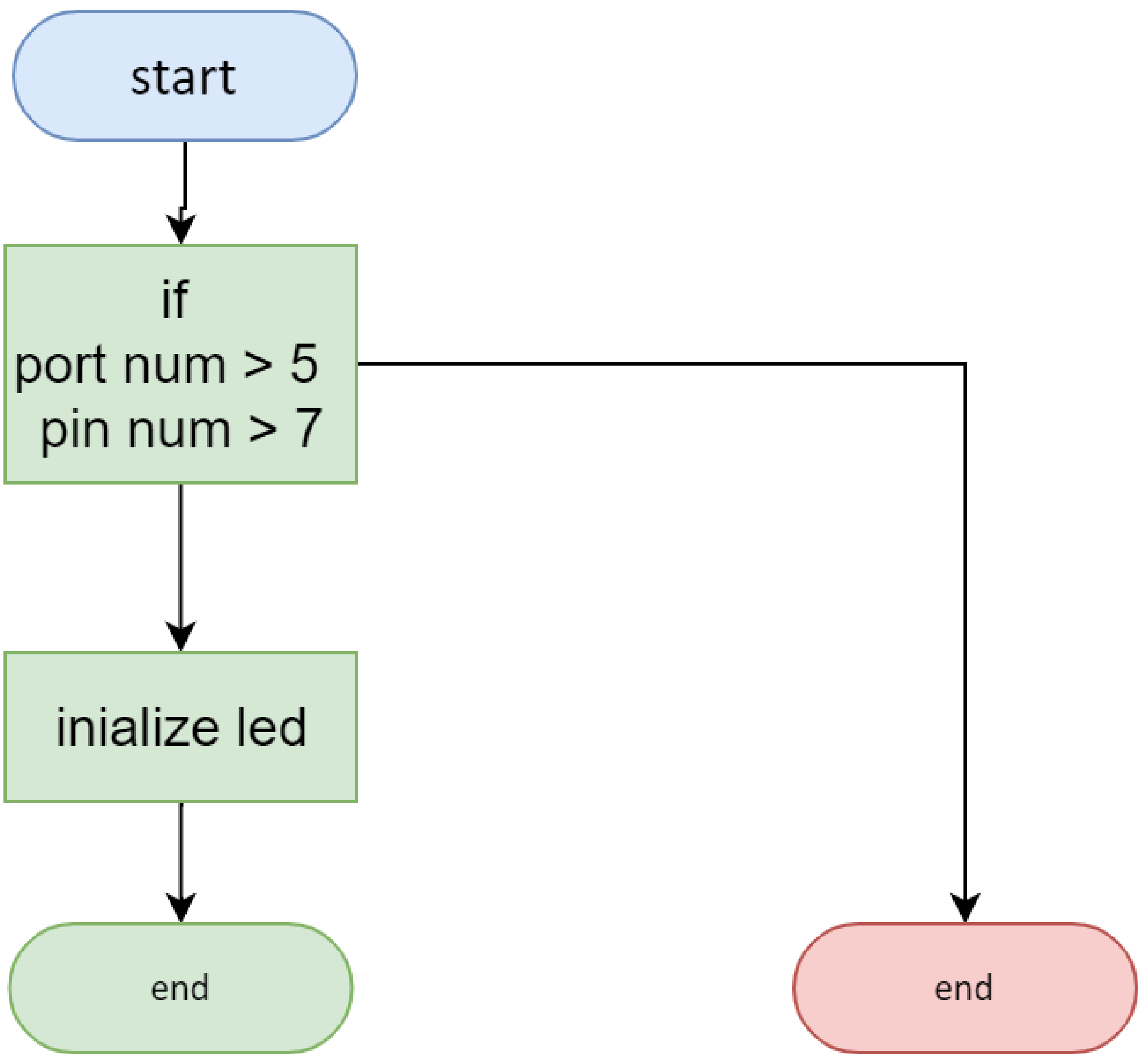
3.2. HAL Layer

3.2.1. LED Module

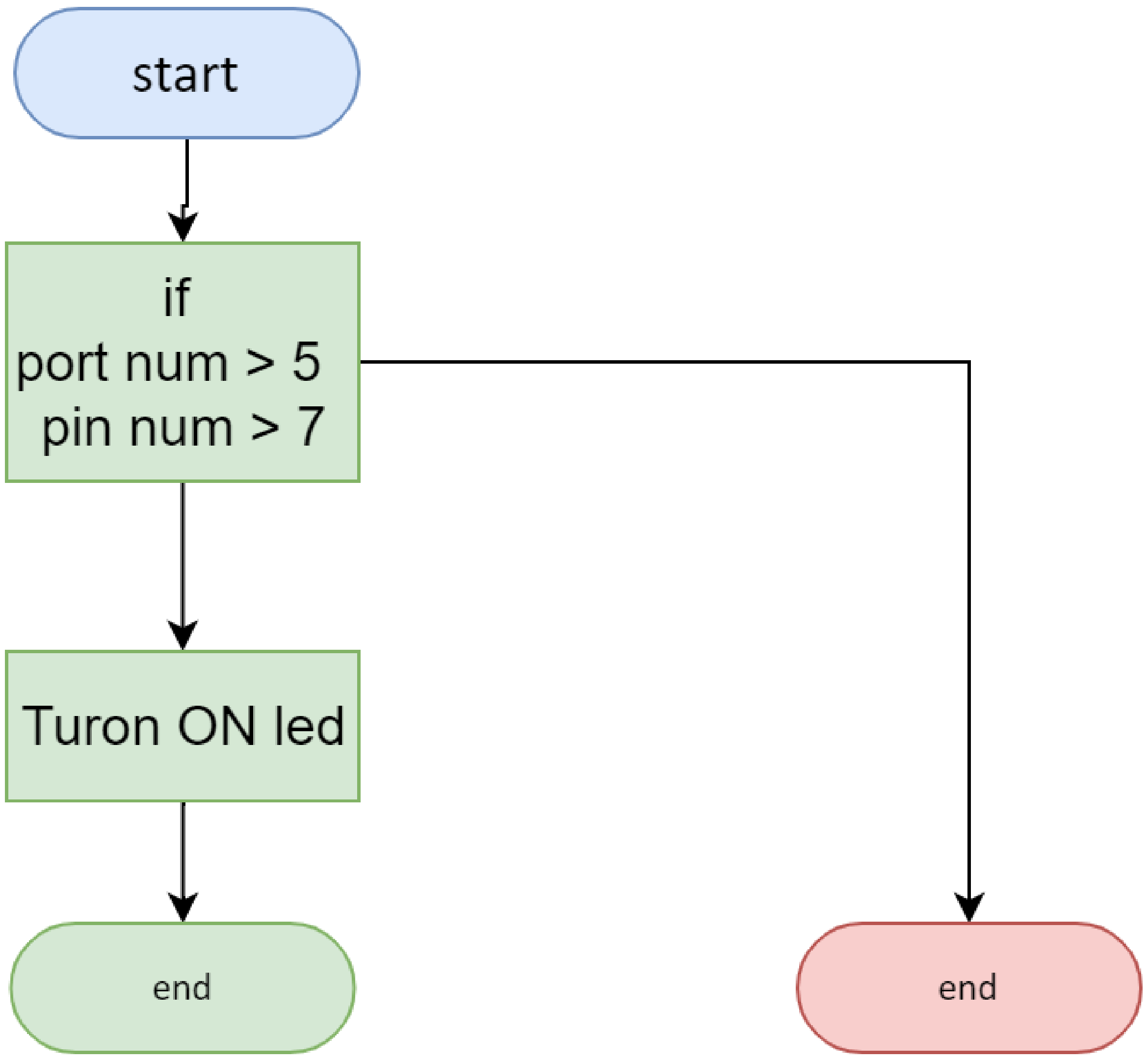
3.2.1.1. Led_init



3.2.1.2. Led_on

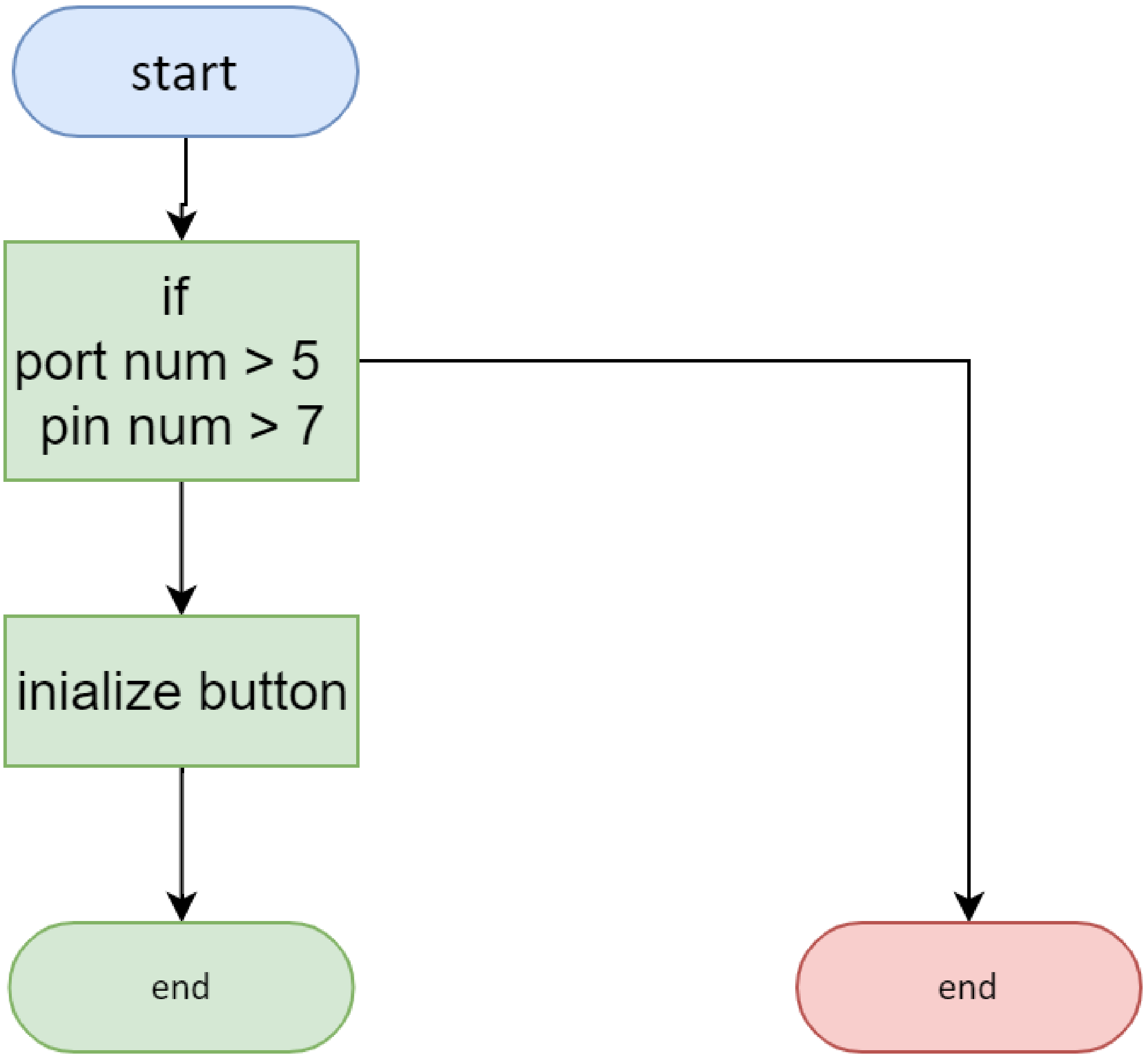


3.2.1.3. Led_off

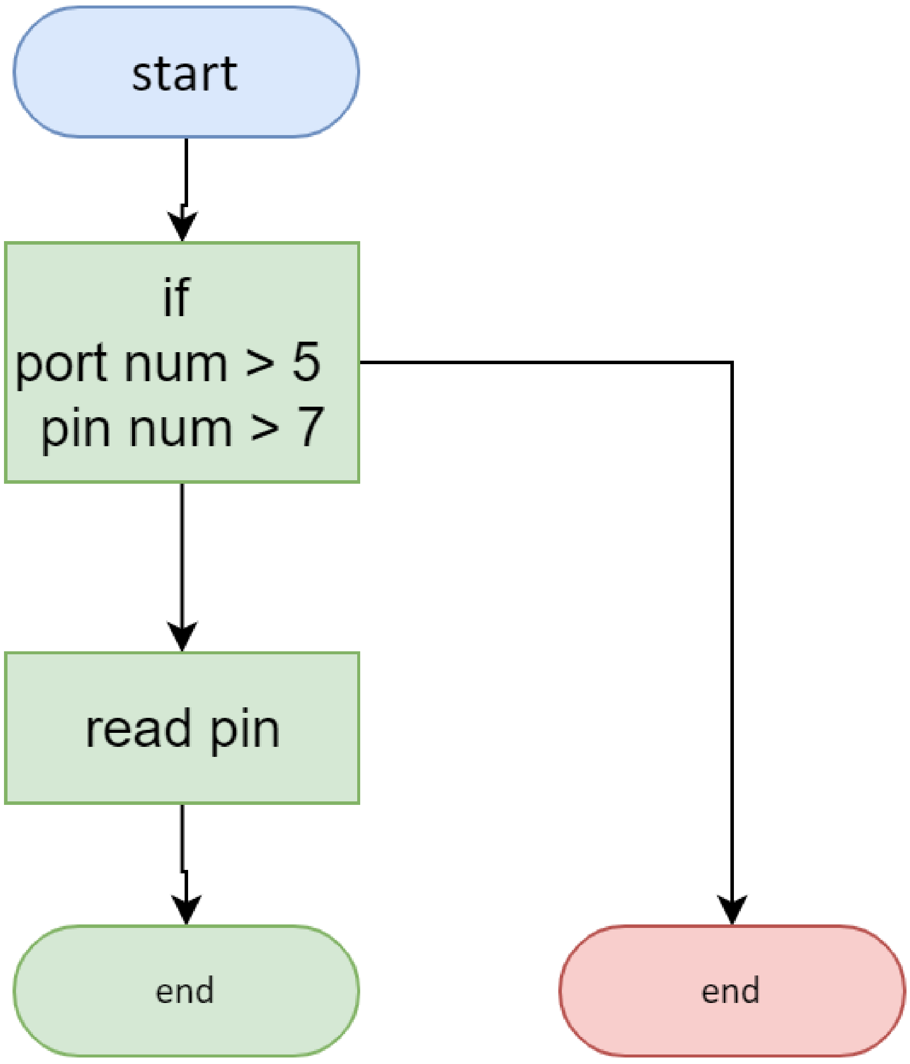


3.2.2. BTN Module

3.2.2.1. Button_init



3.2.2.2. Button_get_state


start
if
port num > 5
pin num > 7
read pin
end
end