Design document

project title: RGB LED Control V1.0 Design

Represented by: Hazem Ashraf

Team: 3

Table of contents

• Project Description

You are supposed to develop the GPIO Driver and use it to control RGB LED on the TivaC board based using the push button.

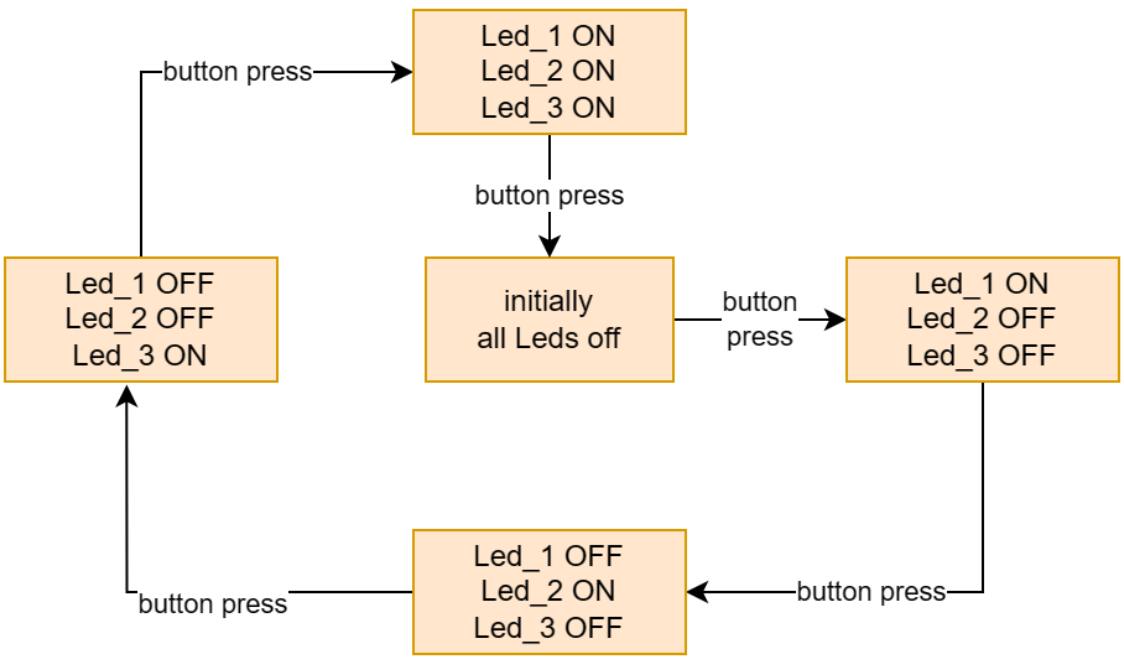
• Project components

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

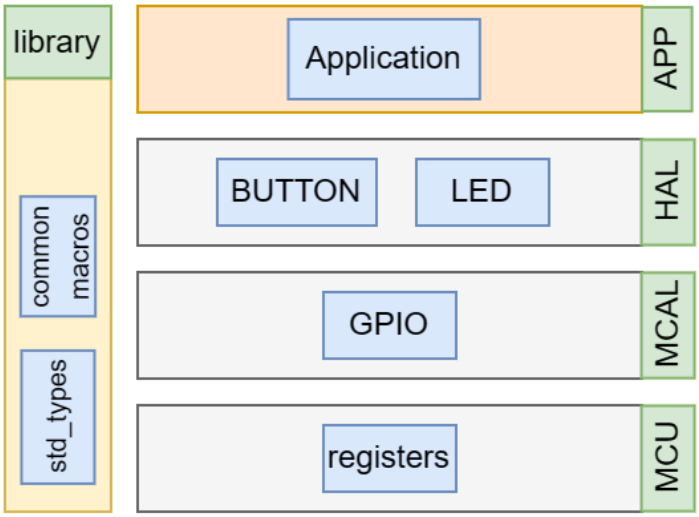
• Main Application Flow

The RGB LED is OFF initially
Pressing SW1:

After the first press, the Red led is on
After the second press, the Green Led is on
After the third press, the Blue led is on
After the fourth press, all LEDs are on
After the fifth press, should disable all LEDs
After the sixth press, repeat steps from 1 to 6



- 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.

  - **Library**

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

  - **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

- **APP APIs**

1- app_init function will initialize the button and led

Function Name	**app_init**
Syntax	**void app_init (void);**
Sync/Async	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/Async	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/Async	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/Async	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/Async	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/Async	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/Async	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/Async	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/Async	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

- **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/Async	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/Async	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**

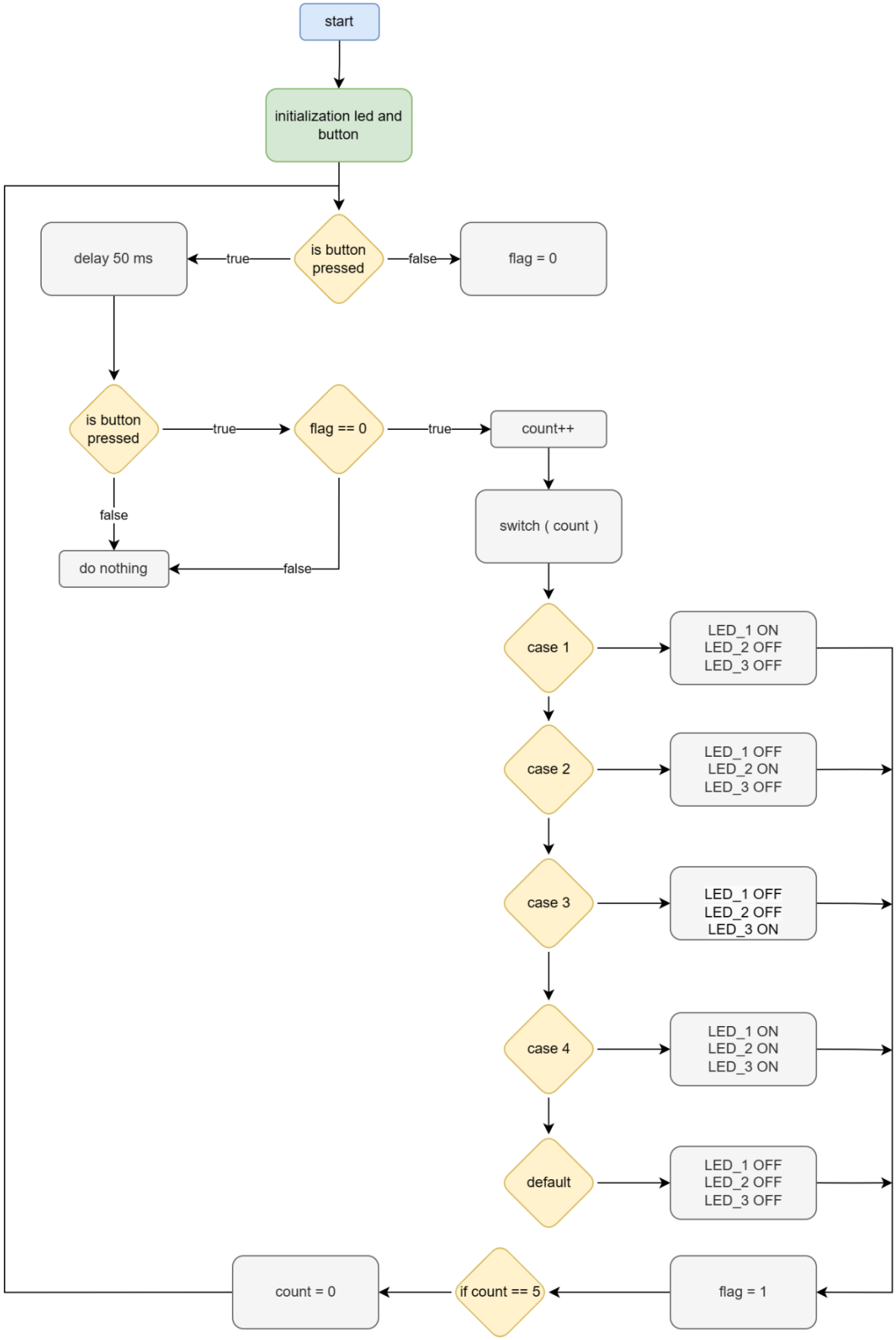
1- BUTTON_init function will initialize button

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/Async	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

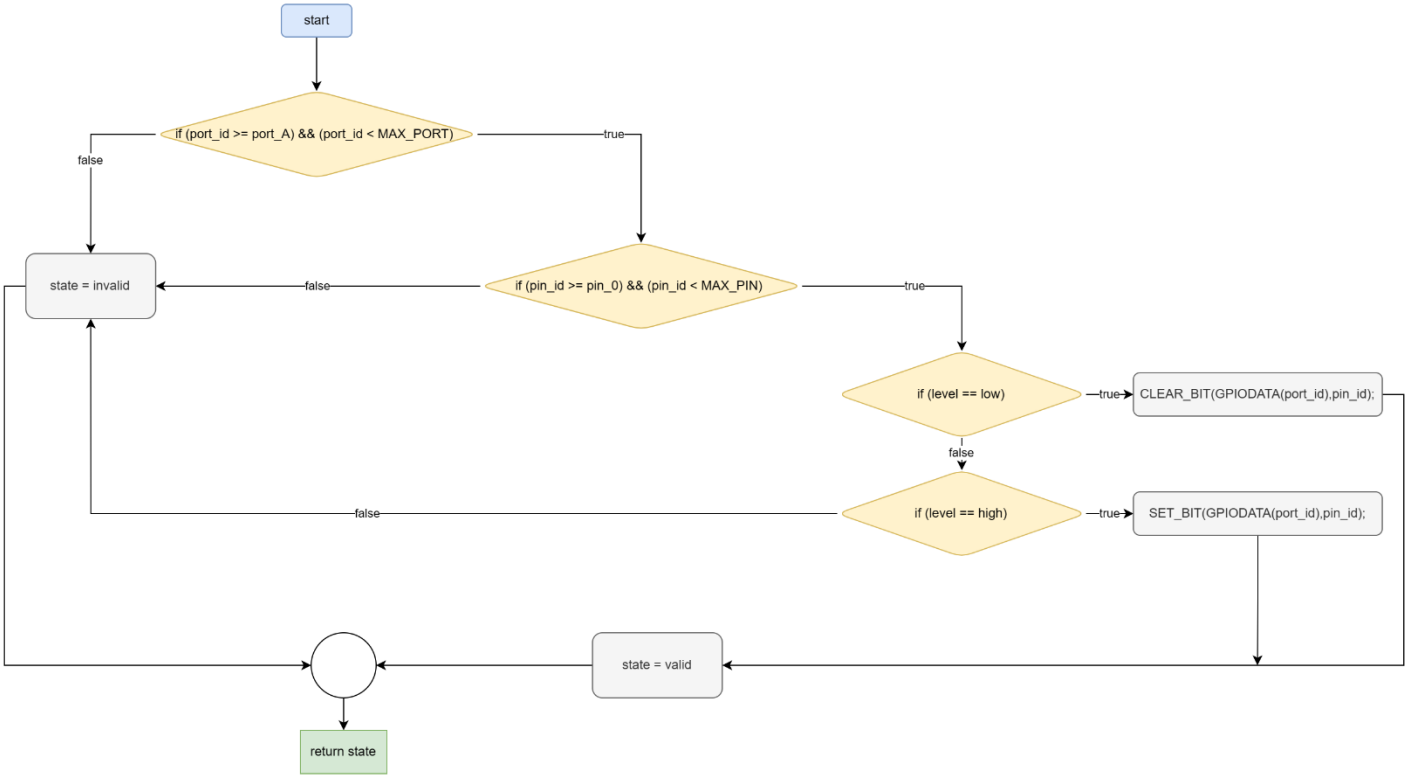
Function Name	BUTTON_digitalRead
Syntax	enu_error_state_t BUTTON_digitalRead (enu_gpio_port_id_t enu_gpio_port_id,enu_gpio_pin_id_t enu_gpio_pin_id,uint8* p_value);
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in):	**enu_gpio_port_id :** port id should be one of the following [ENU_PORT_A, 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

App flowchart

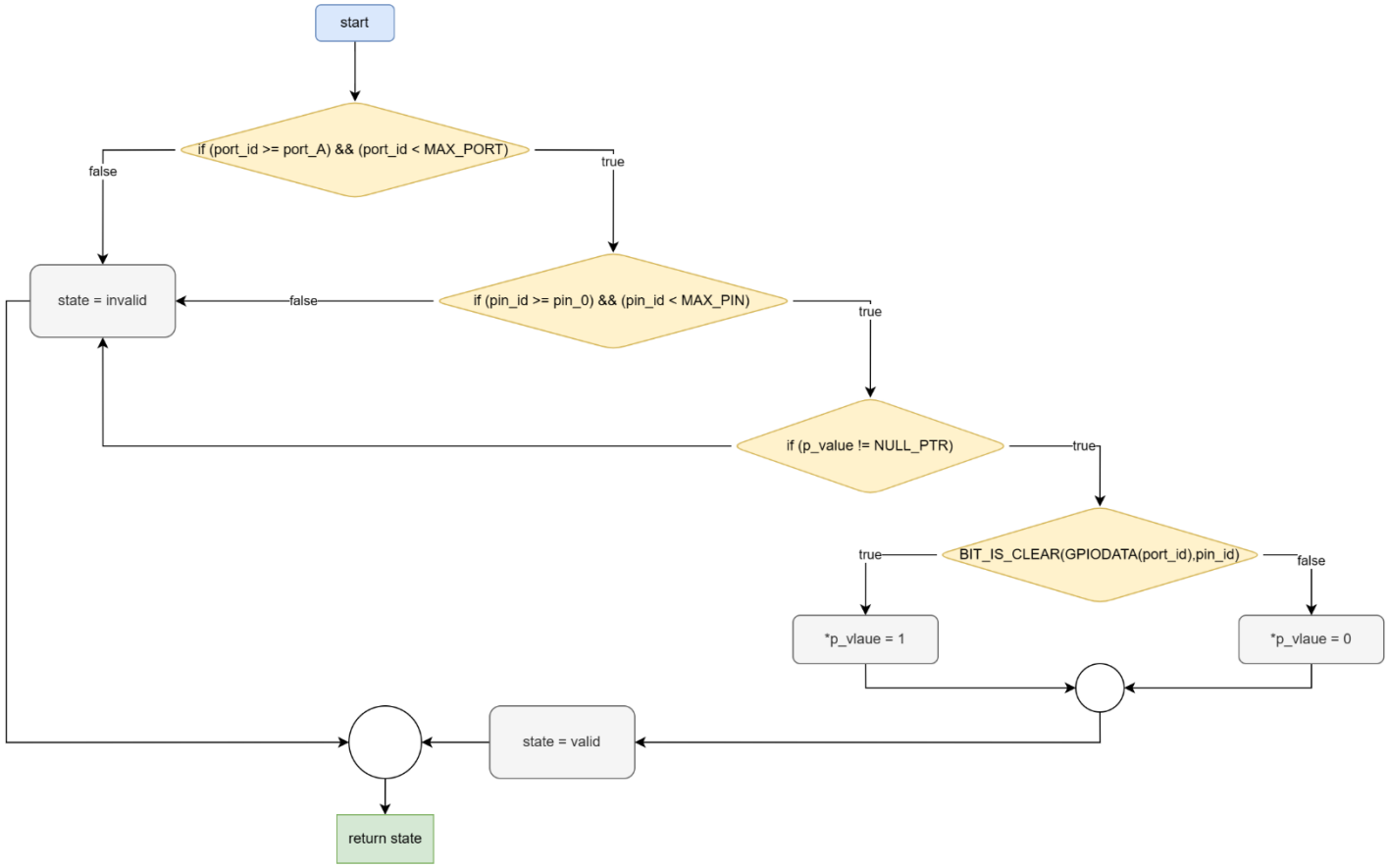


GPIO flowchart

- **GPIO_digitalWrite**

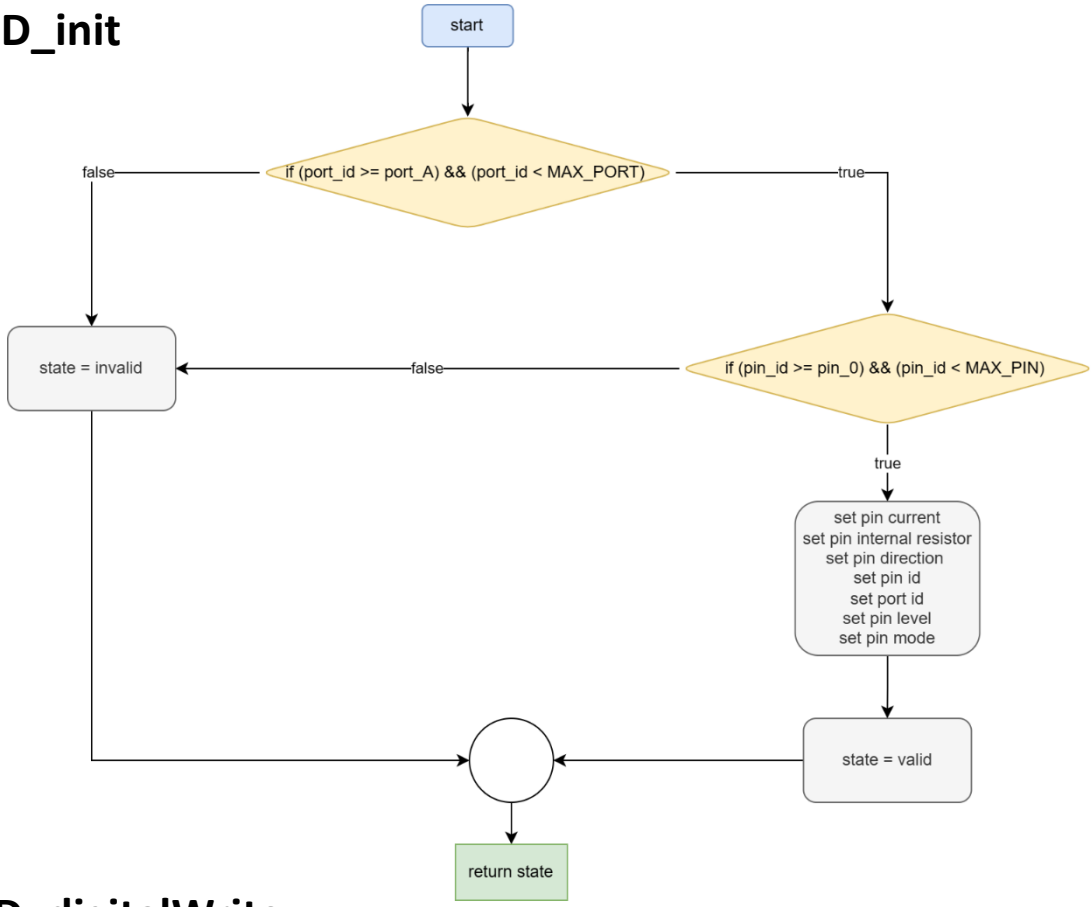


- **GPIO_digitalRead**

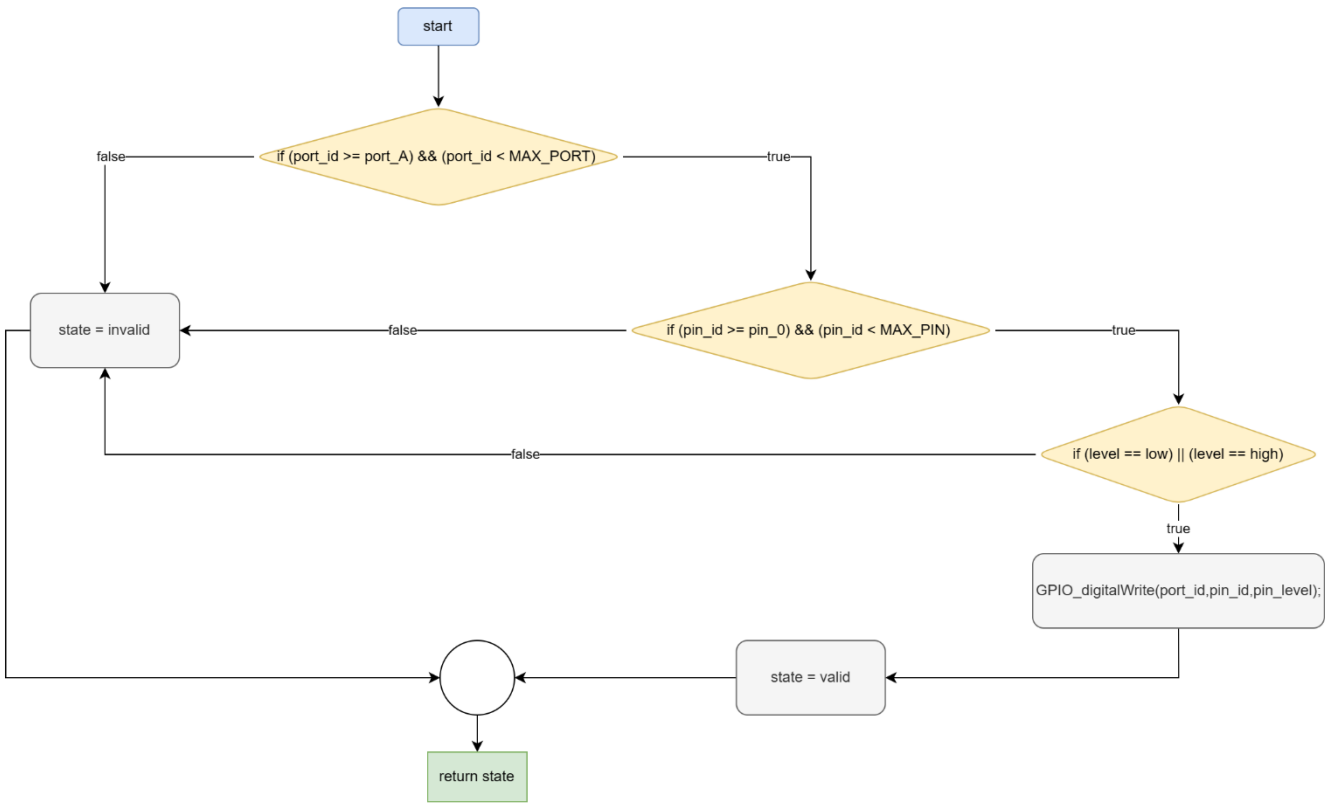


LED flowchart

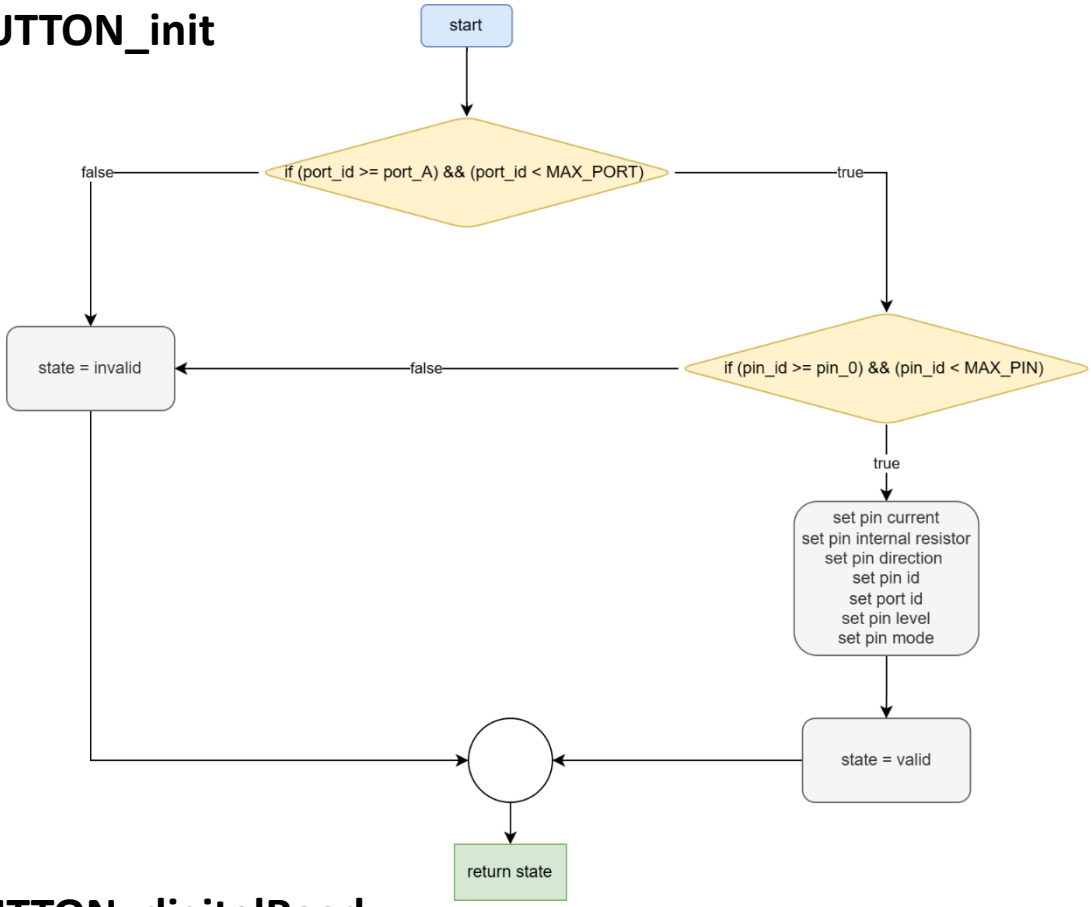
- **LED_init**



- **LED_digitalWrite**



- **BUTTON_init**



- **BUTTON_digitalRead**