DESIGN DOCUMENT

RGB LED CONTROL

V3.0 DESIGN

Team2:

-Amr El-Abd

-Ahmed Aatef

Table of Contents: -

Project introduction:

This project aims to implement a system using the TivaC board, where the SW1 button and RGB LED are utilized. The system operates in four states, controlled by the number of times SW1 is pressed. Initially, the RGB LED is turned off, and when SW1 is pressed for the first time, the Green LED turns on with a 30% duty cycle. With subsequent presses of SW1, the duty cycle increases to 60% and then 90%. Upon the fourth press, the Green LED turns off. Additionally, on the fifth press, the system state returns to the first state. To achieve this functionality, the project requires the implementation of a General Purpose Timer (GPT) driver, which supports various APIs such as initialization, timer control, interrupt handling, and callbacks.

Project description :-

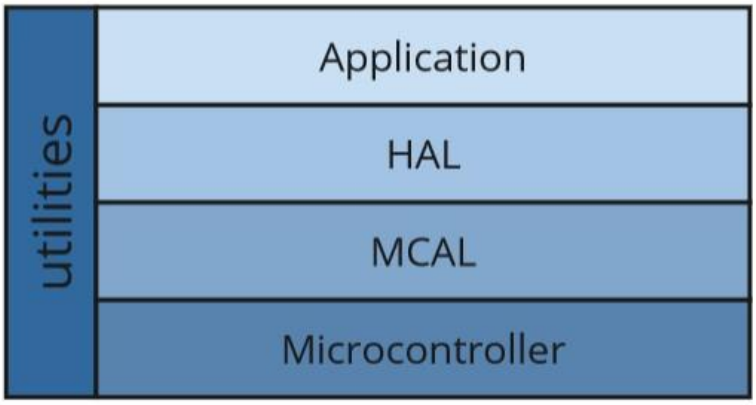
1. Hardware Requirements

1. Use the TivaC board.
2. Use SW1 as an input button.
3. Use the RGB LED.

2. Software Requirements

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

Layered Architectures:-



Application Layer: This is the topmost layer of the software stack, which contains the actual application logic. It interacts with the lower layers to perform its tasks. It is responsible for implementing the desired functionality of the system.

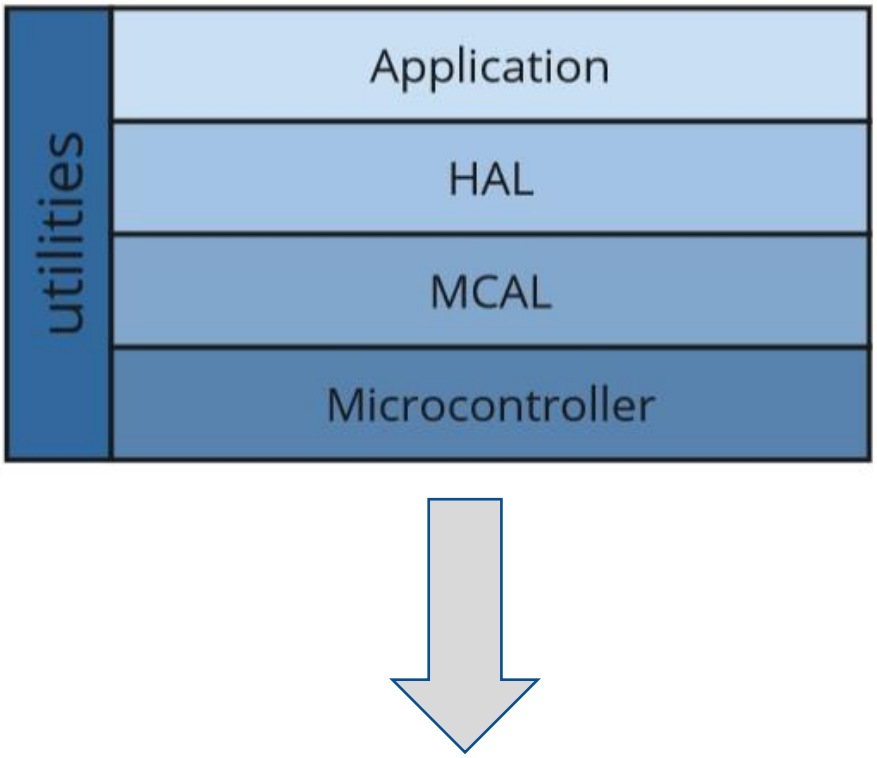
HAL Layer: This layer provides an abstraction for external devices connected to the microcontroller. The HAL layer provides interface to access external devices and hides the implementation details from the application layer.

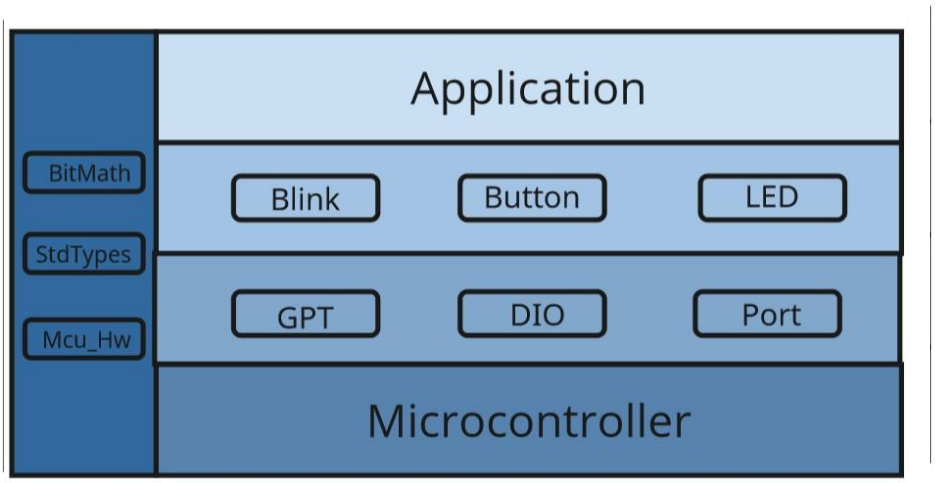
MCAL Layer (Microcontroller Abstraction Layer): This layer provides an abstraction for the microcontroller hardware. It includes low-level drivers for peripherals. It hides the hardware details and provides a uniform interface to the upper layers.

Utilities Layer: the utilities layer includes memory mapping, standard types, and utils.h. Memory mapping involves defining the memory layout and addresses for different components. Standard types provide a set of predefined data types that ensure consistency and portability across different platforms. The utils.h header file contains utility functions and macros that offer commonly used functionalities, such as bit manipulation.

Microcontroller: This layer represents the physical hardware layer consisting of the microcontroller chip. The microcontroller is responsible for executing the code stored in its memory and controlling the behavior of the system.

System modules:-





Drivers' documentation:-

Port driver:

Description: Driver to Setup the pin configuration:
- Setup the pin as Digital GPIO pin
- Setup the direction of the GPIO pin
- Set the passed initial values for the GPIO pin
- Setup the mode of the GPIO pin
- Setup the internal resistor for i/p pin
- Setup the output current in case of output pi

APIs:

```
enu_ErrorReturn PortInit(const strPortConfig_t* ConfigPtr);
```

DIO driver:

Description: Driver for DIO to read/write/toggle Channel

APIs:

```
enuDioLevel_t DioReadChannel(DioChannel_t ChannelId);

enu_ErrorReturn DioWriteChannel(DioChannel_t ChannelId, enuDioLevel_t Level);

enuDioLevel_t DioToggleChannel(DioChannel_t ChannelId);
```

LED driver:

Description: Driver to initialize/ turn on/ turn off/ toggle the connected channel

APIs:

```
Enu_ErrorReturn  LedInit(void);

Enu_ErrorReturn  LedTurnOn(LedChannel_t LedChannel);

Enu_ErrorReturn  LedTurnOff(LedChannel_t LedChannel);

Enu_ErrorReturn  LedToggle(LedChannel_t LedChannel);
```

Button driver:

Description: Driver to initialize and get the state of the connected Buttons

APIs:

Enu_ErrorReturn ButtonInit(void);

enuButtonState_t ButtonGetState(ButtonChannel_t ButtonChannel, enuButtonAttach_t ButtonAttach);

GPT driver:

Description: this driver provides an interface for controlling and utilizing general-purpose timers on the TivaC board, offering functions such as timer initialization, start/stop operations, time measurement, interrupt handling, and callback support.

APIs:

Enu_ErrorReturn Gpt_Init(const Gpt_ChannelConfigType* ConfigPtr);

Enu_ErrorReturn Gpt_DisableNotification(Gpt_ChannelType ChannelId);

Enu_ErrorReturn Gpt_EnableNotification(Gpt_ChannelType ChannelId);

Enu_ErrorReturn Gpt_StopTimer(Gpt_ChannelType ChannelId);

Gpt_ValueType Gpt_GetTimeElapsed(Gpt_ChannelType ChannelId);

Gpt_ValueType Gpt_GetTimeRemaining(Gpt_ChannelType ChannelId);

Std_ReturnType Gpt_GetPredefTimerValue(Gpt_PredefTimerType PredefTimer, uint32* TimeValuePtr);

Blink driver:

Description: this driver provides the ability to blink the required LED channels with the required duty cycle depending on the GPT driver APIs and configuration.

APIs:

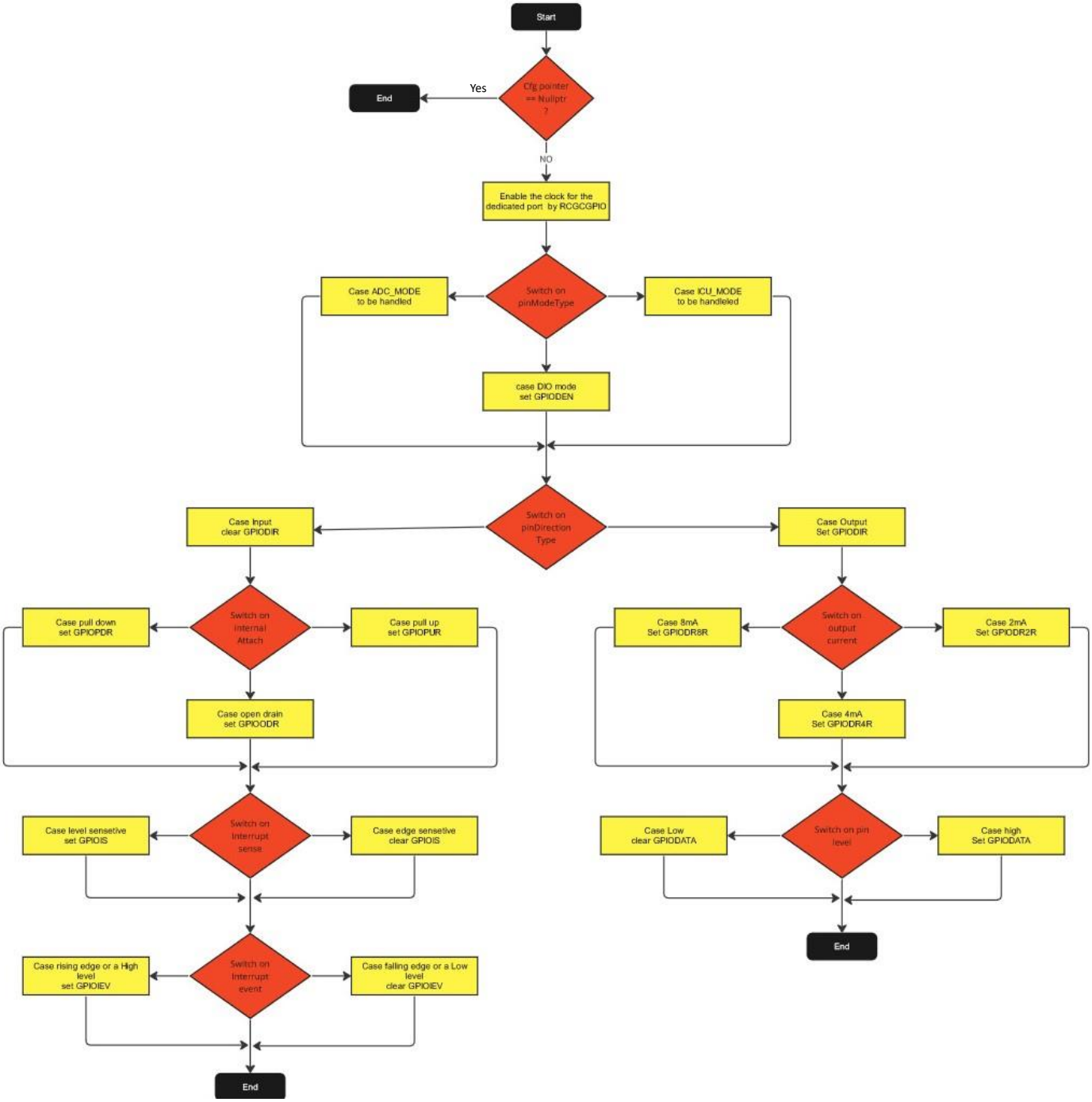
enu_ErrorReturn blink_init (Service_TimerChannelType TimerChannle, Service_DeviceChannel DeviceChannel);

enu_ErrorReturn Blink_Start2(Service_TimerChannelType TimerChannle, uint16 Time, uint16 HighPeriod, uint16 LowPeriod);

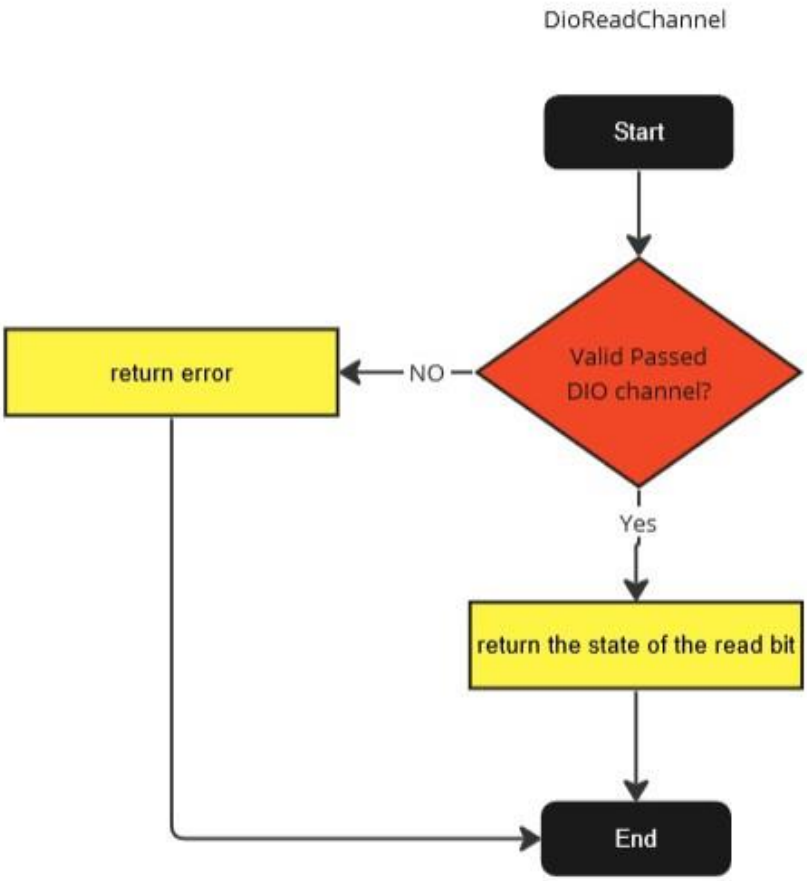
Flowcharts for Functions:

Port driver:

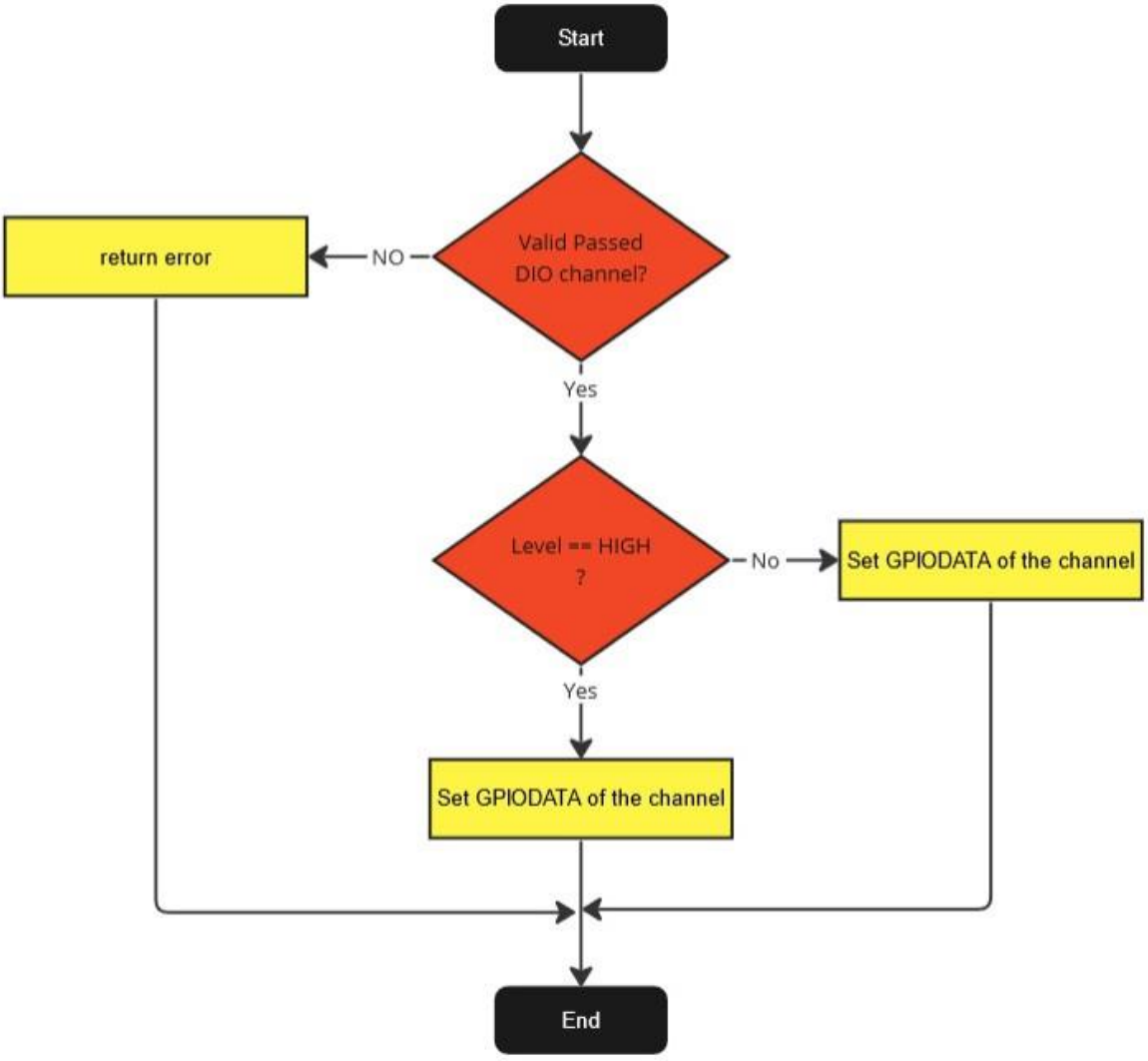
Port init :



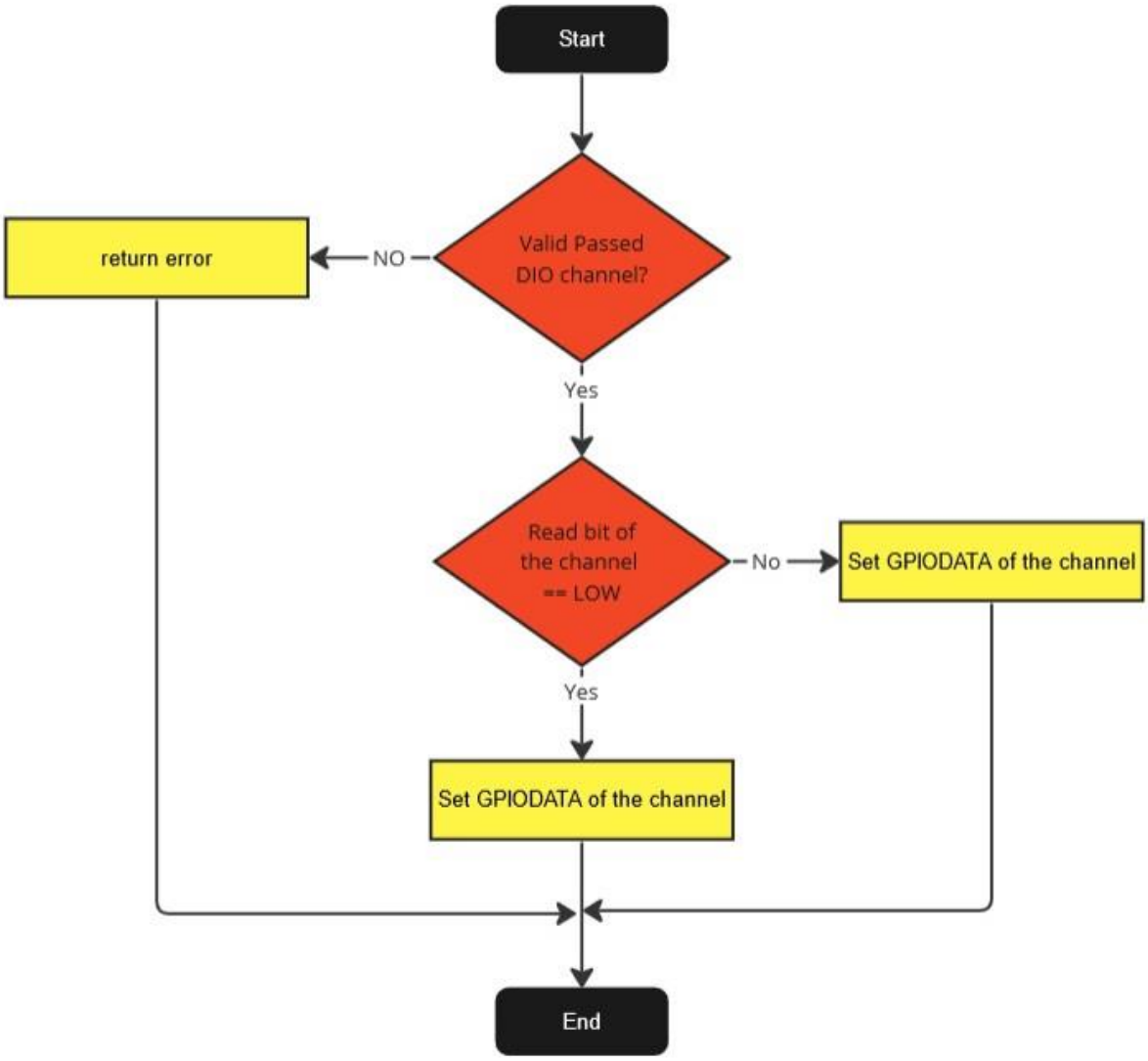
DIO driver:


DioReadChannel
Start
Valid Passed DIO channel?
NO
return error
Yes
return the state of the read bit
End

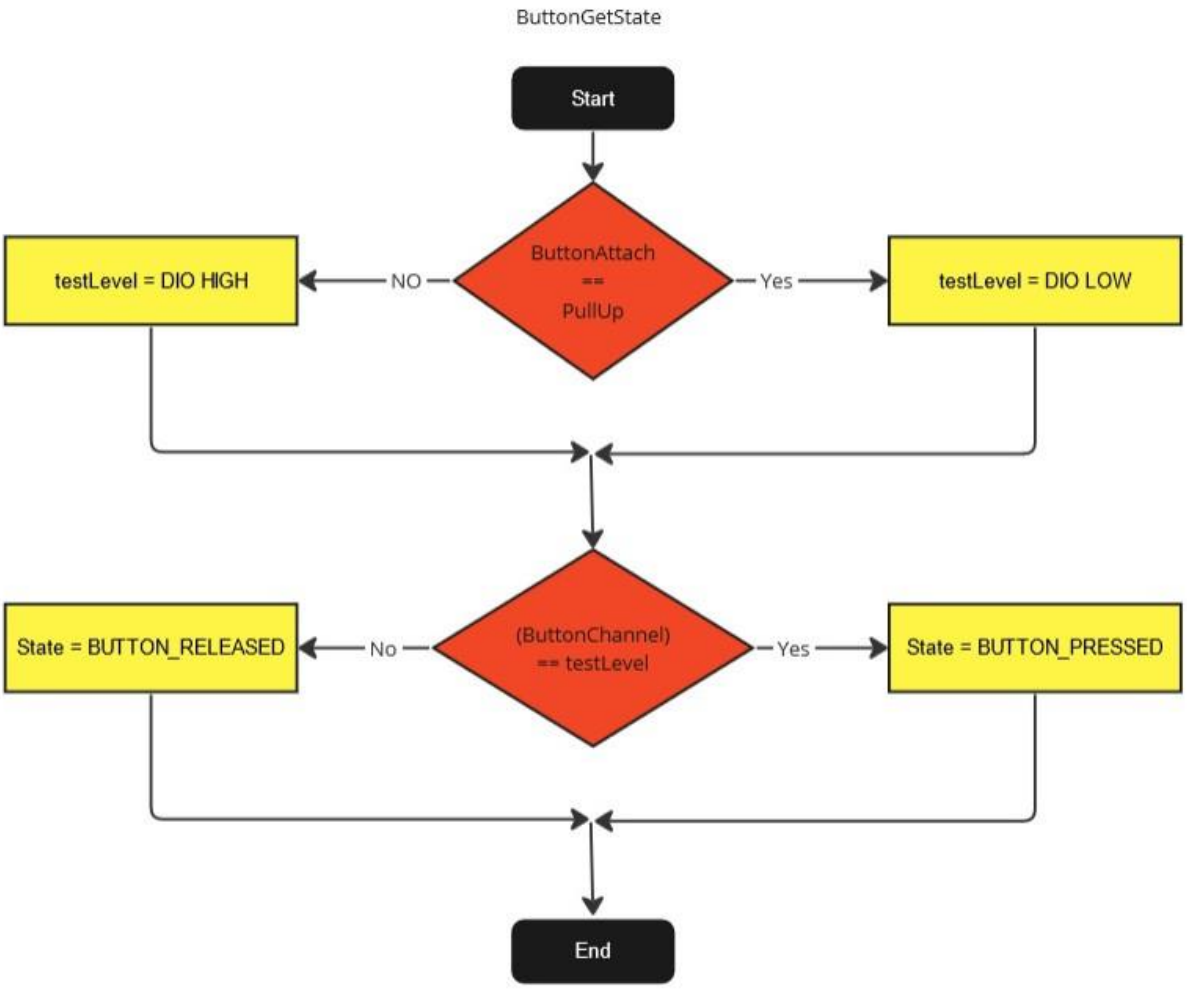
DioWriteChannel

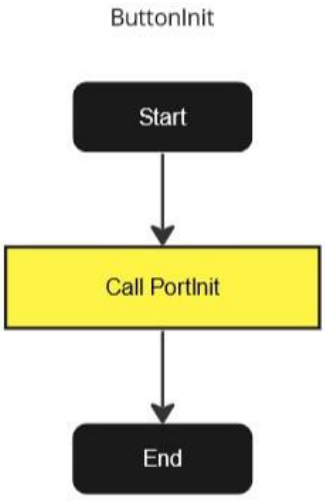


DioToggleChannel

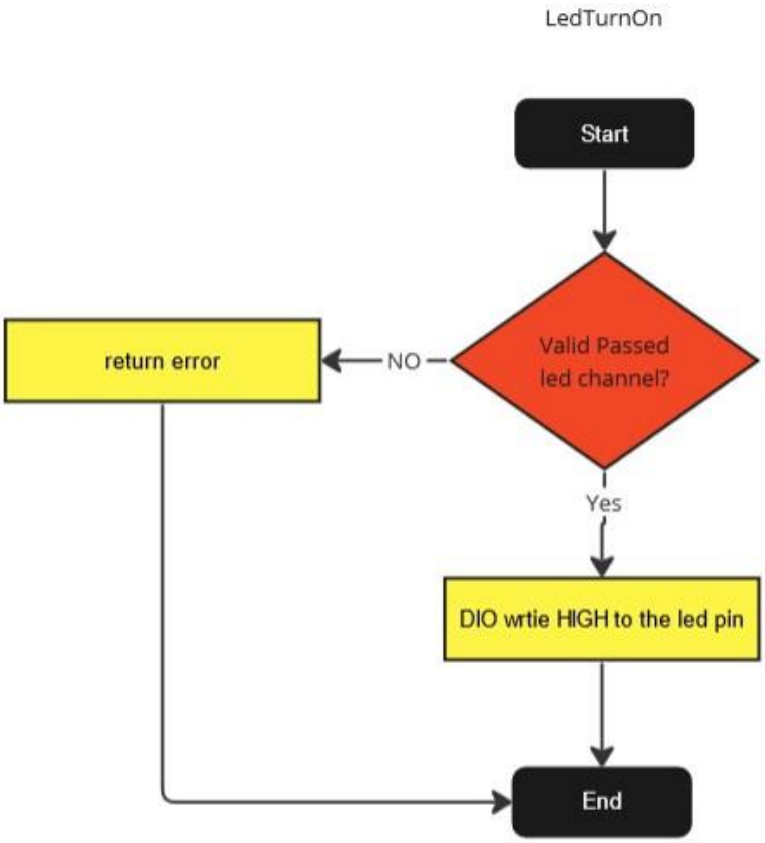


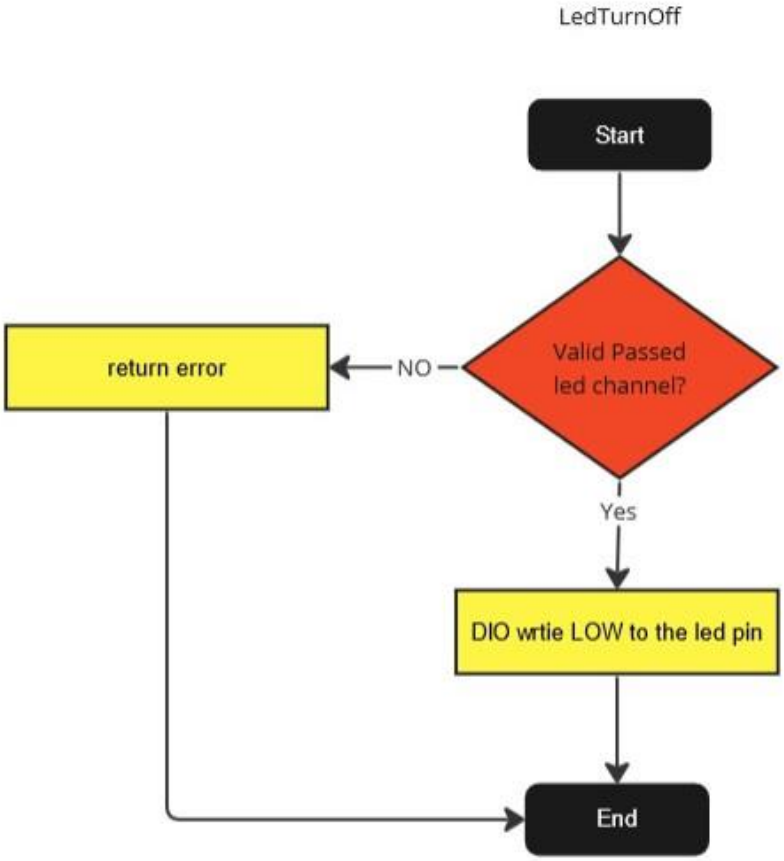
Button driver



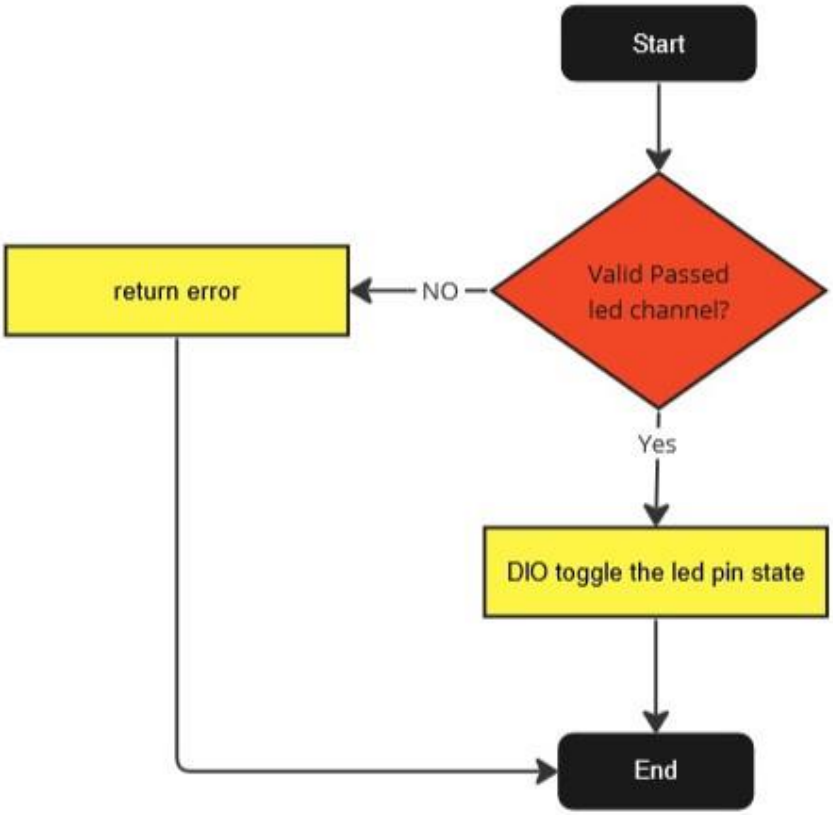


LED driver:

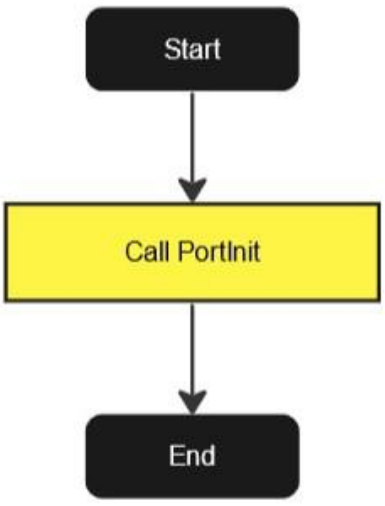

LedTurnOn
Start
Valid Passed
led channel?
NO
return error
Yes
DIO wrtie HIGH to the led pin
End


LedTurnOff
Start
Valid Passed
led channel?
NO
return error
Yes
DIO wrtie LOW to the led pin
End

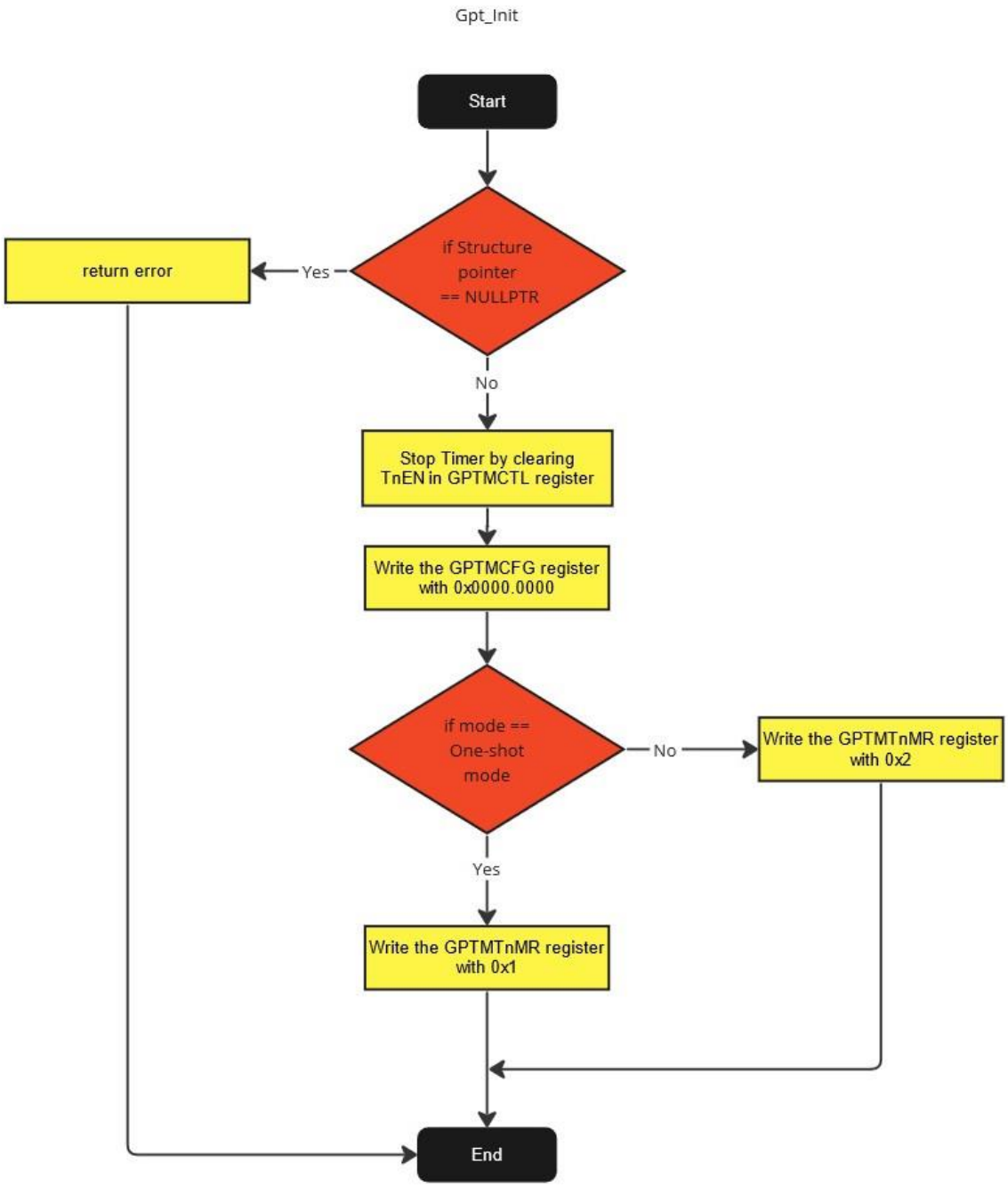
LedToggle



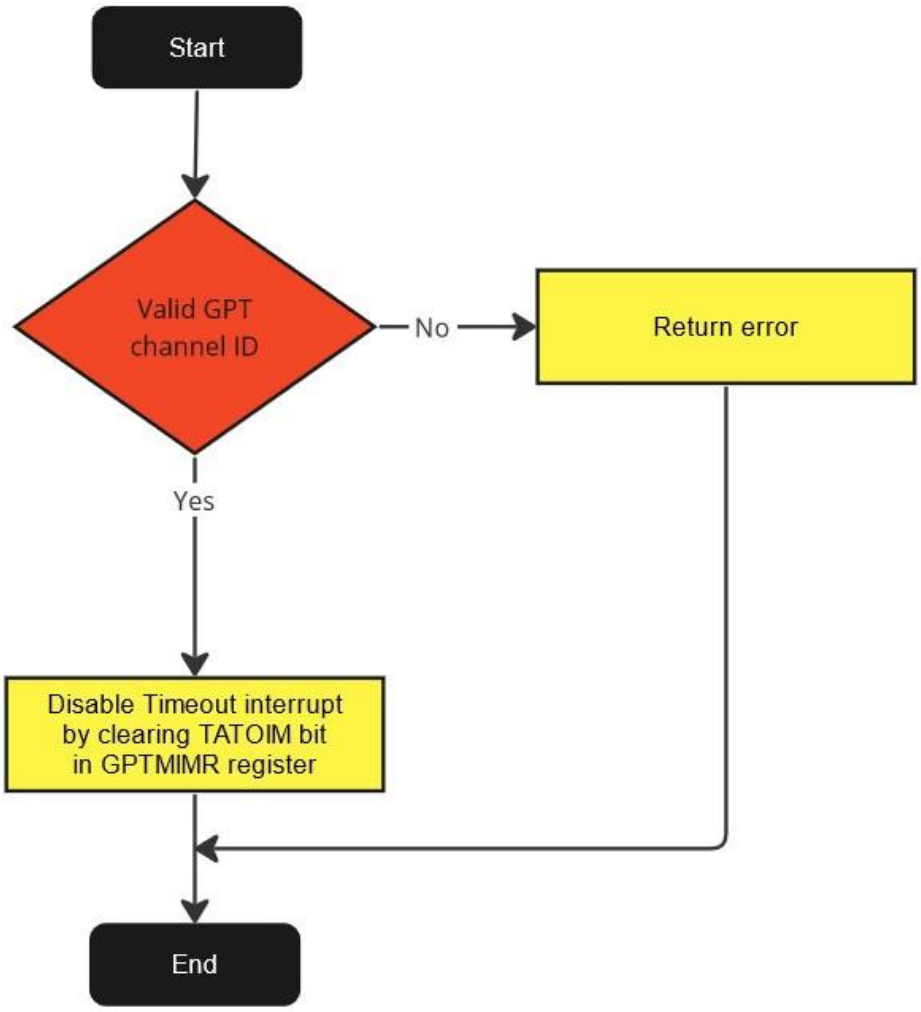
LedInit



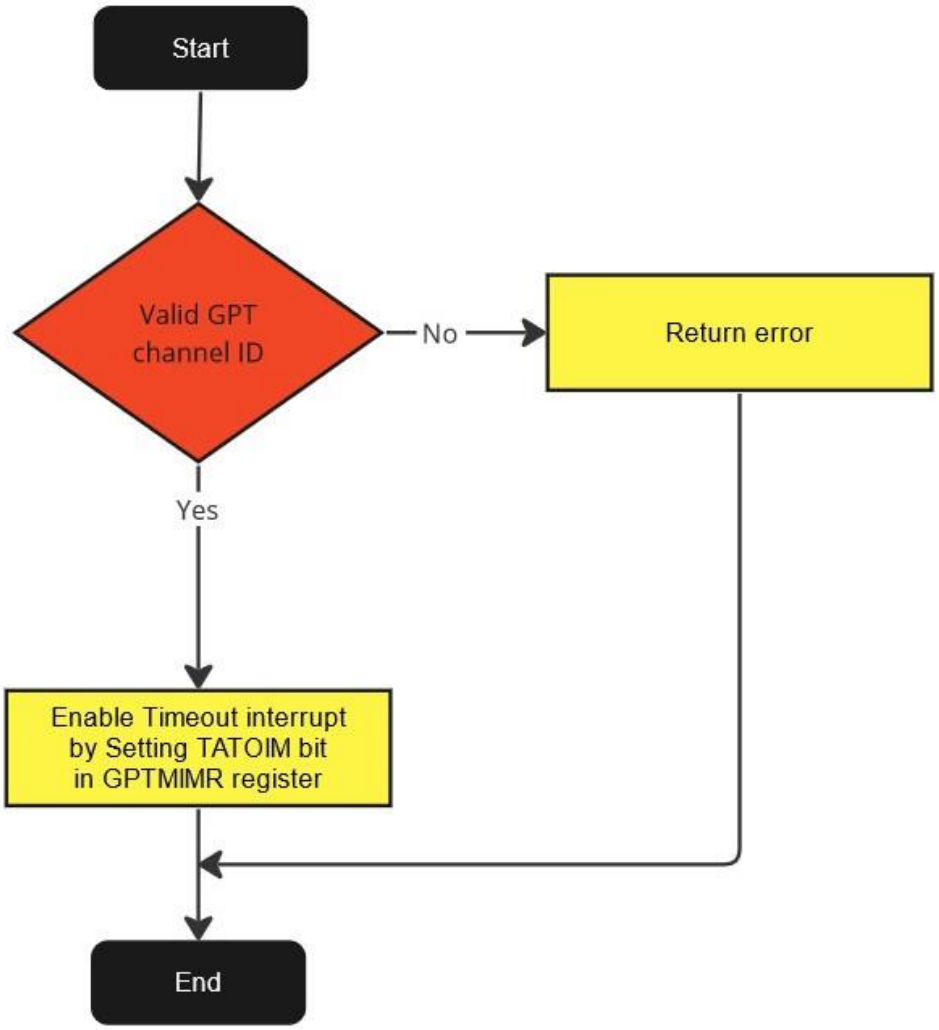
GPT Functions:-



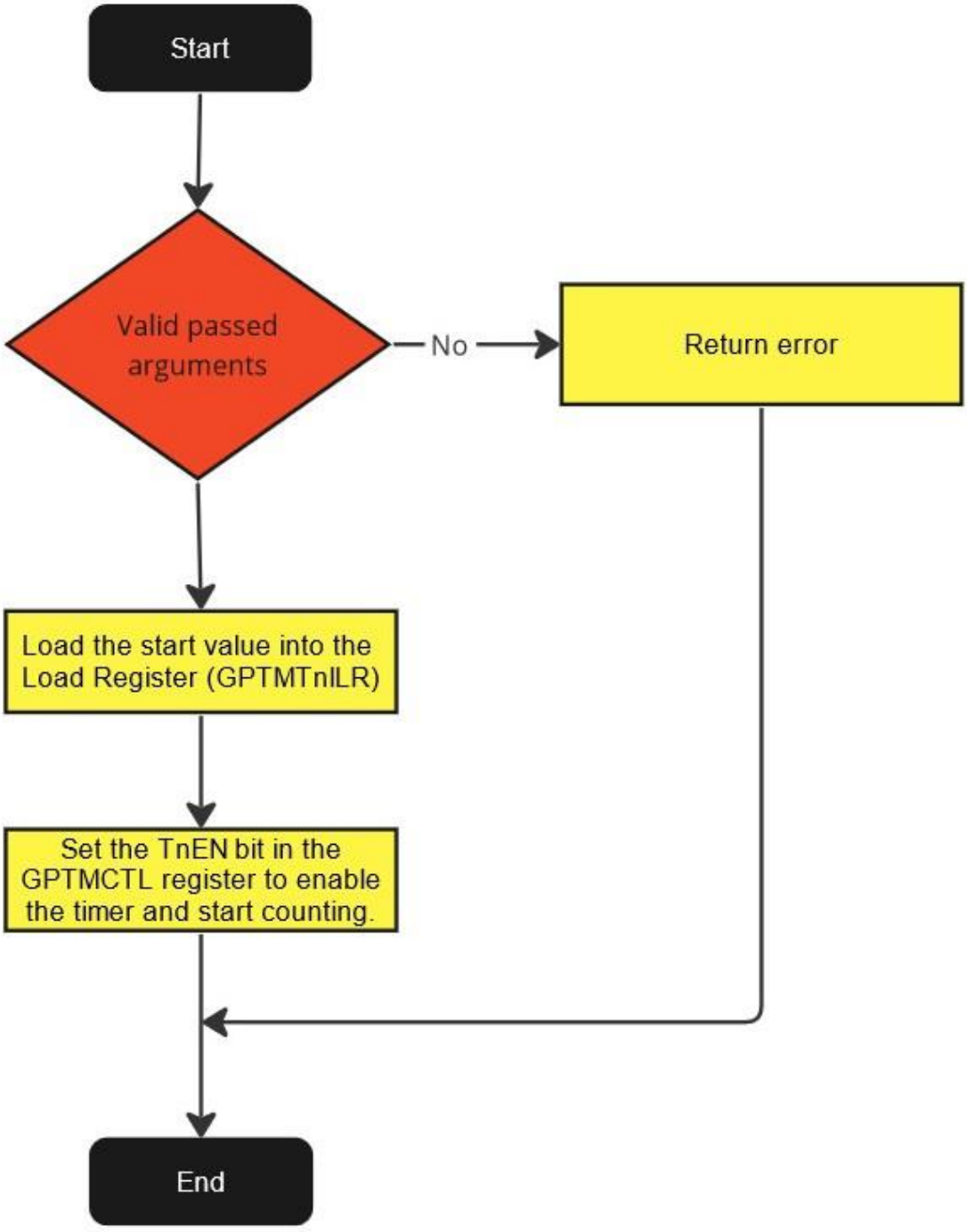
Gpt_DisableNotification



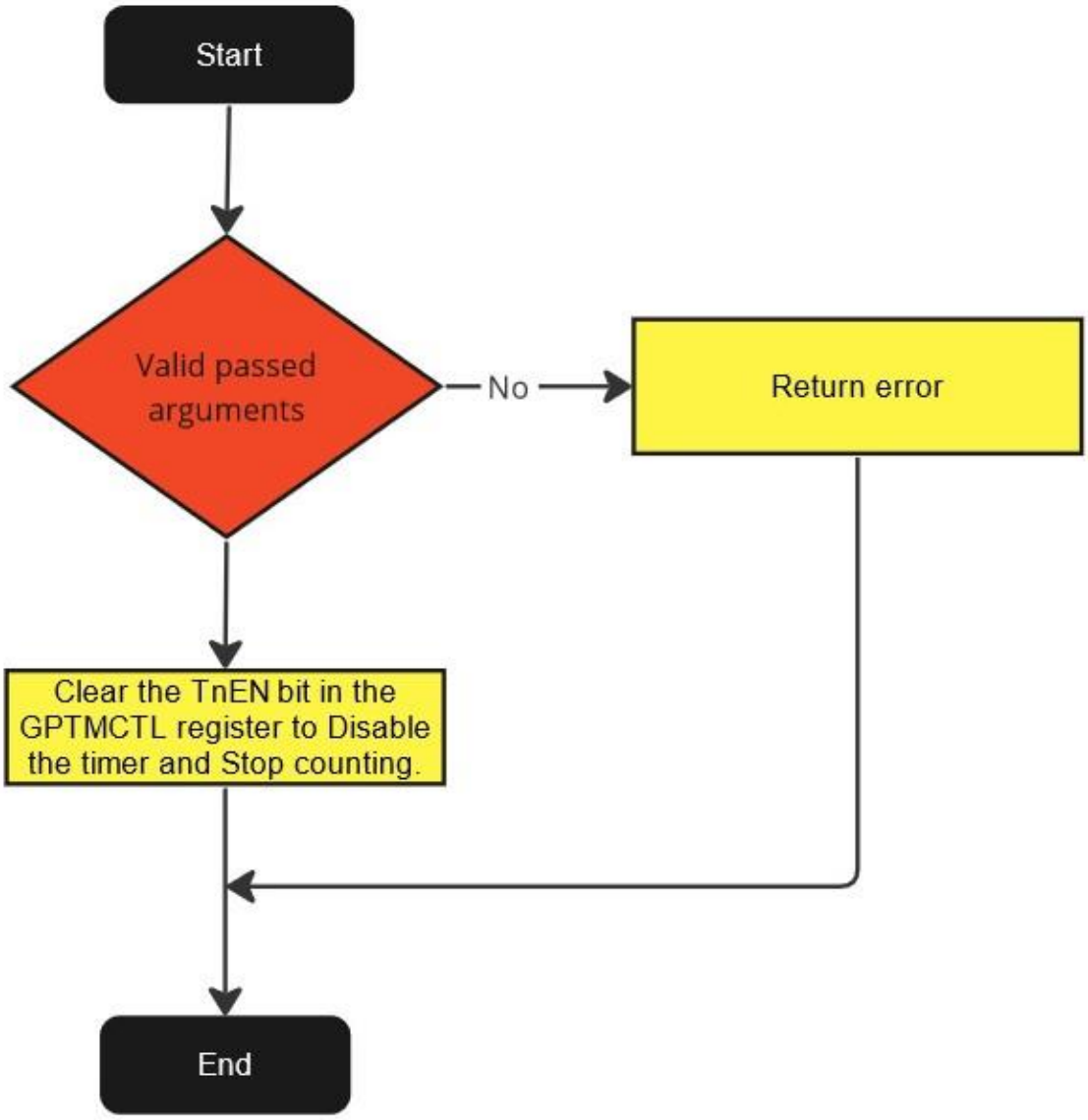
Gpt_EnableNotification



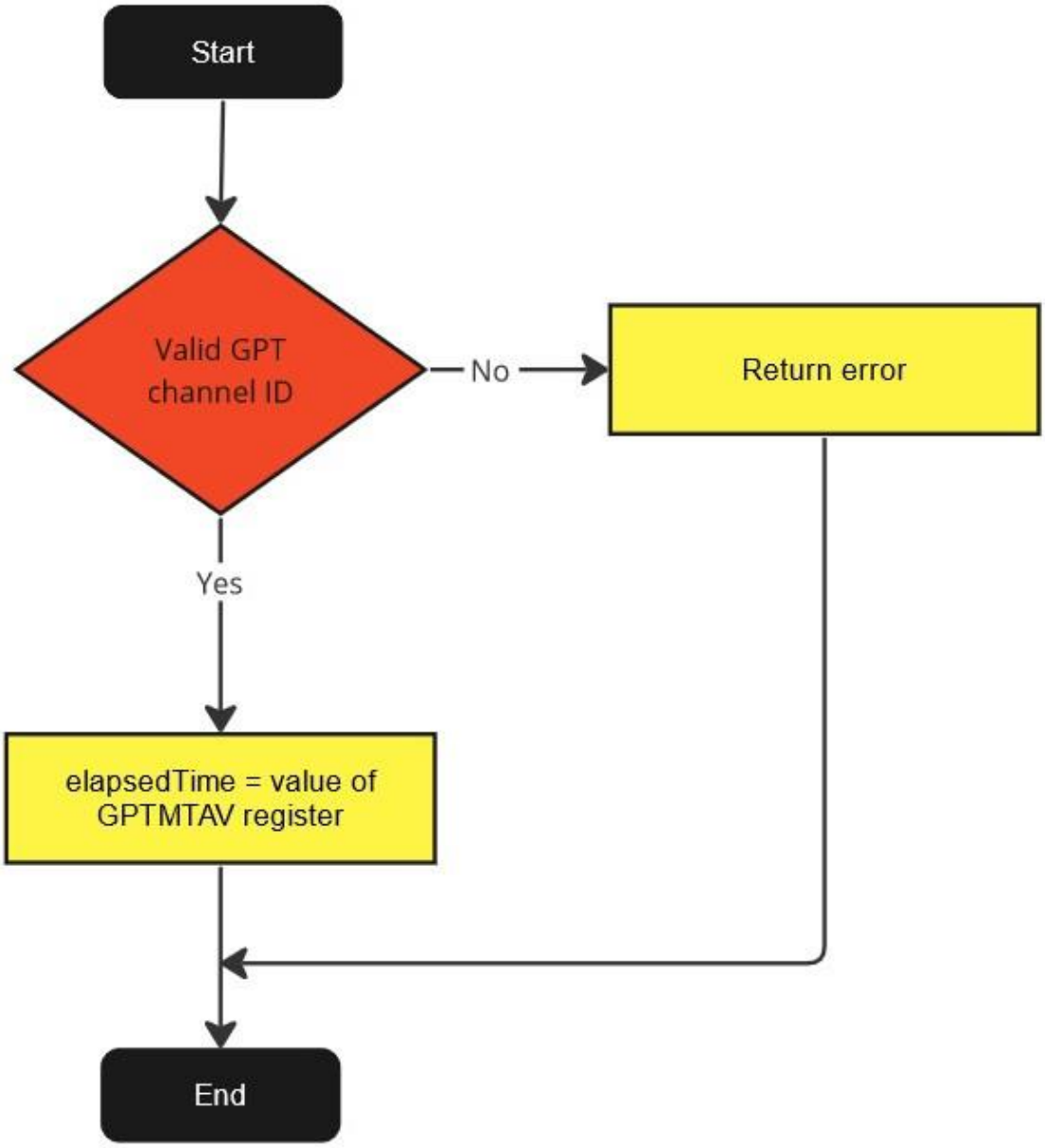
Gpt_StartTimer



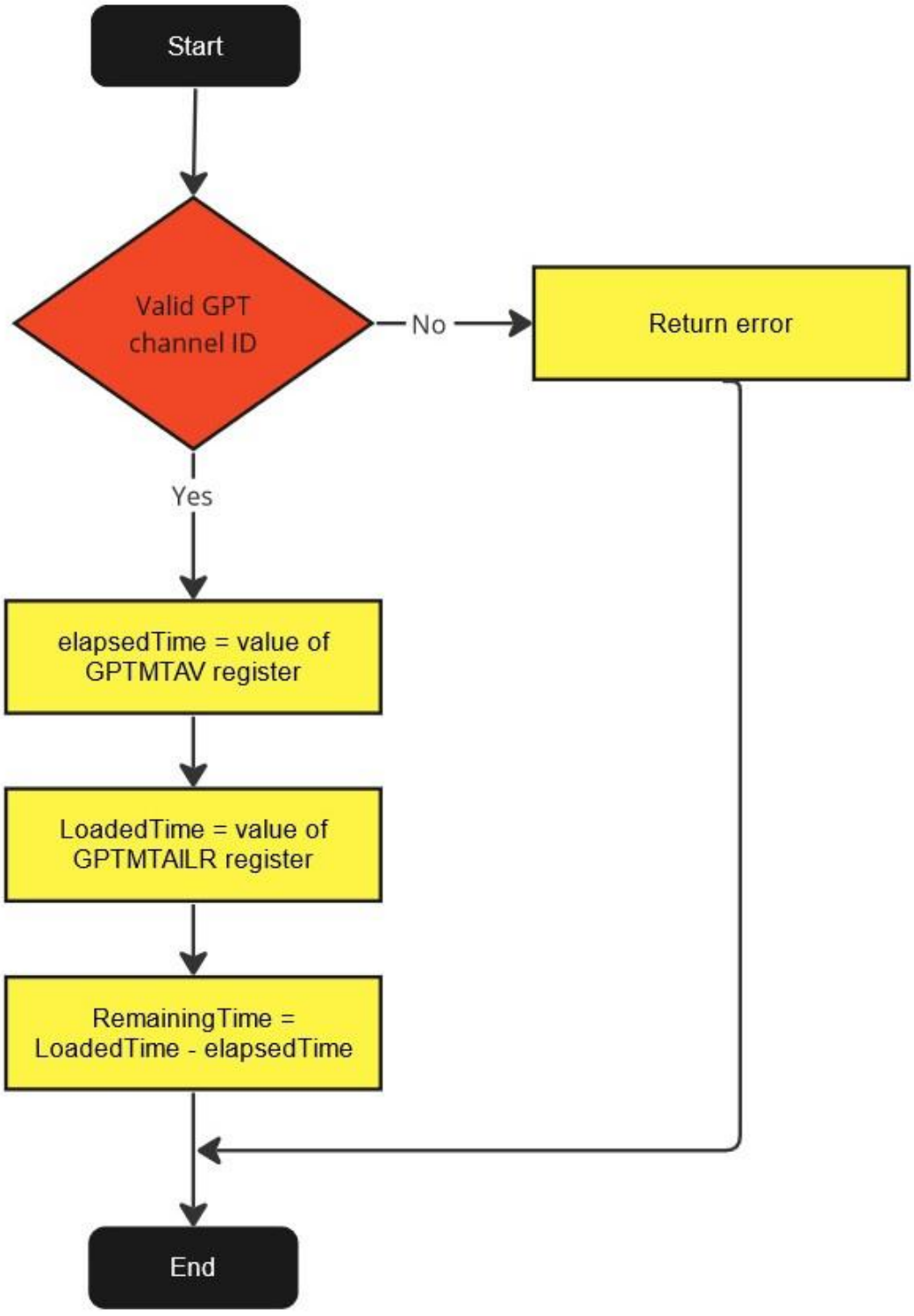
Gpt_StopTimer



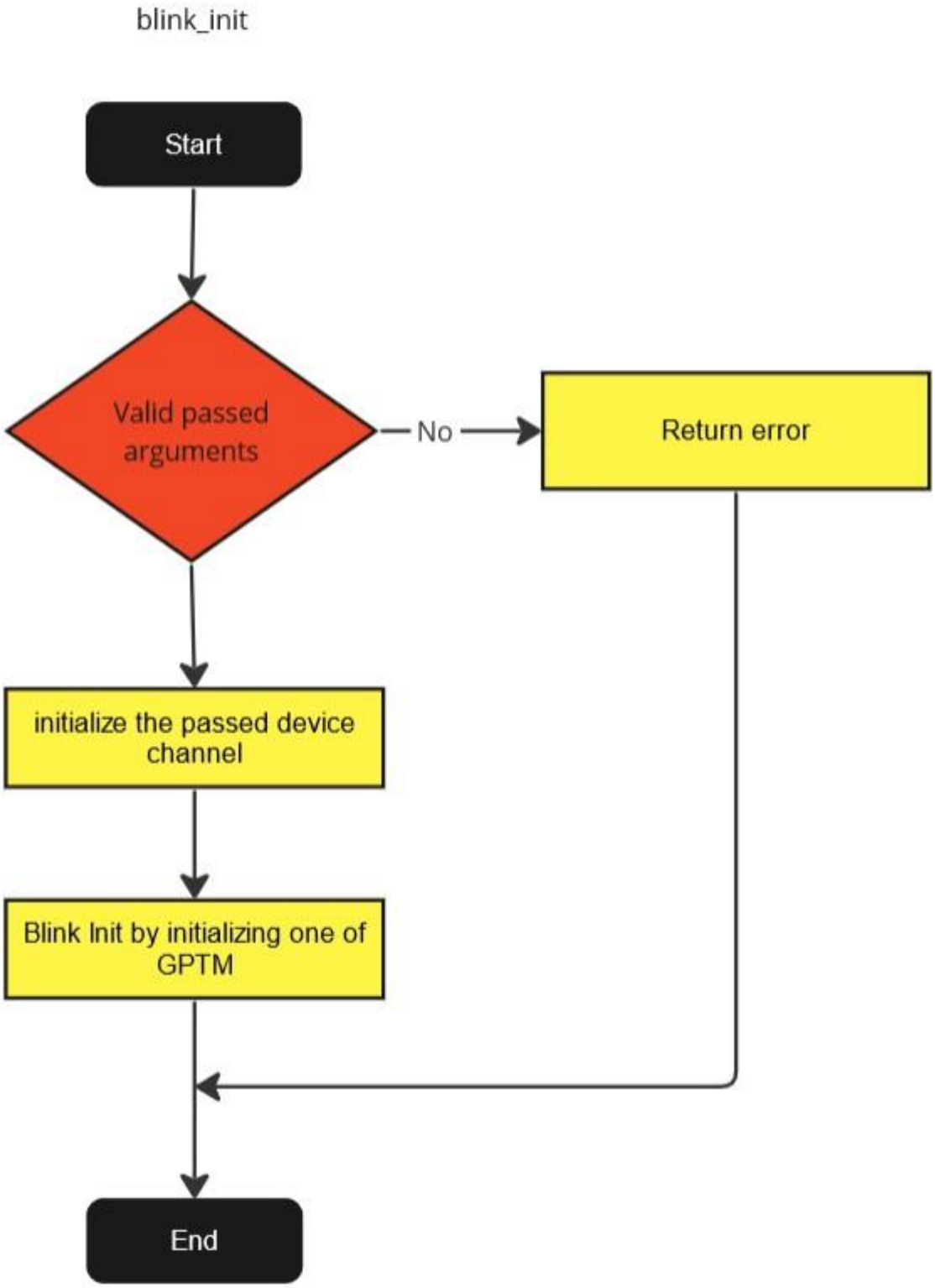
Gpt_GetTimeElapsed

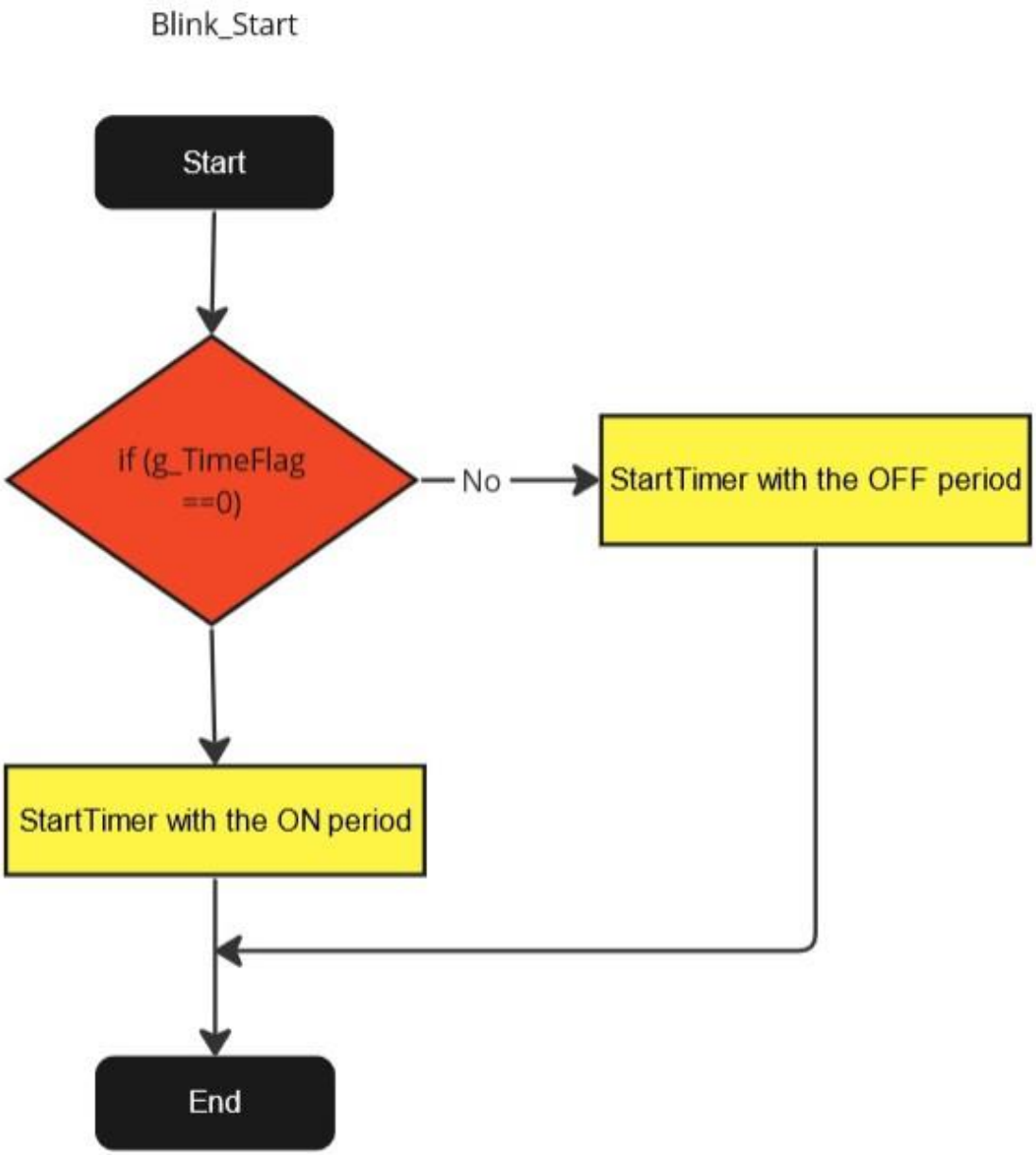


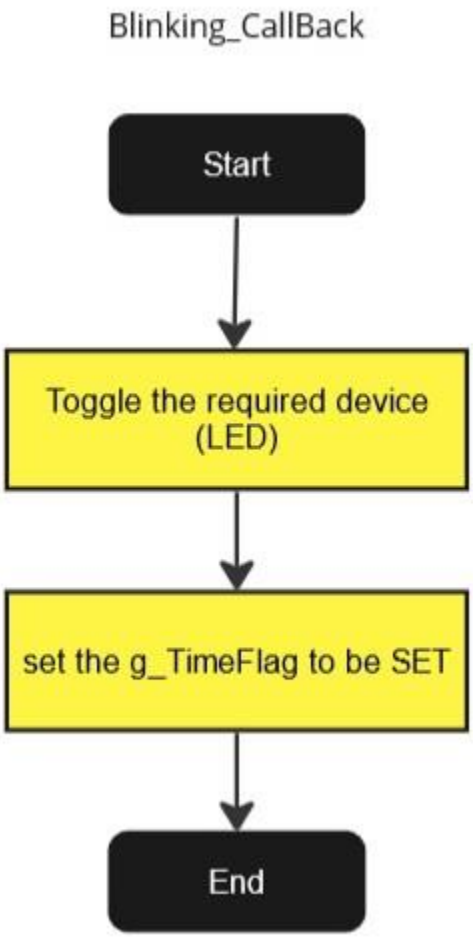
Gpt_GetTimeRemaining



Blink functions:




Blink_Start
Start
if (g_TimeFlag ==0)
No
StartTimer with the OFF period
StartTimer with the ON period
End


Blinking_CallBack
Start
Toggle the required device (LED)
set the g_TimeFlag to be SET
End

Precompiling and linking configurations :-

Port Lcfg:

```
const strPortConfig_t strPortConfig =
{
    /********************************** PORTA ***********************************/
    PORTA, PIN0, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,
    PORTA, PIN1, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,
    PORTA, PIN2, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,
    PORTA, PIN3, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,
    PORTA, PIN4, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,
    PORTA, PIN5, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,
    PORTA, PIN6, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,
    PORTA, PIN7, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

    /********************************** PORTB ***********************************/
```

PORTB, PIN0, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTB, PIN1, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTB, PIN2, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTB, PIN3, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTB, PIN4, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTB, PIN5, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTB, PIN6, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTB, PIN7, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

/********** PORTC **********/

PORTC, PIN0, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTC, PIN1, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTC, PIN2, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTC, PIN3, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTC, PIN4, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTC, PIN5, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTC, PIN6, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTC, PIN7, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

/********** PORTD **********/

PORTD, PIN0, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTD, PIN1, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTD, PIN2, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTD, PIN3, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTD, PIN4, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTD, PIN5, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTD, PIN6, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTD, PIN7, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

/********** PORTE **********/

PORTE, PIN0, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTE, PIN1, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTE, PIN2, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTE, PIN3, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

PORTE, PIN4, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

```
    PORTE, PIN5, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,Port_IntDisable,

    /********************************** PORTF ***********************************/
    PORTF, PIN0, CHANNEL_ENABLED, INPUT,     PIN_LEVEL_HIGH, DIO_MODE, PULL_UP,       DRIVE_2mA,Port_IntDisable,
    PORTF, PIN1, CHANNEL_ENABLED, OUTPUT, PIN_LEVEL_LOW,  DIO_MODE, PULL_DOWN, DRIVE_2mA,Port_IntDisable,
    PORTF, PIN2, CHANNEL_ENABLED, OUTPUT, PIN_LEVEL_LOW,  DIO_MODE, PULL_DOWN, DRIVE_2mA,Port_IntDisable,
    PORTF, PIN3, CHANNEL_ENABLED, OUTPUT, PIN_LEVEL_LOW,  DIO_MODE, PULL_DOWN, DRIVE_2mA,Port_IntDisable,
    PORTF, PIN4, CHANNEL_ENABLED, INPUT,     PIN_LEVEL_HIGH, DIO_MODE, PULL_UP,       DRIVE_2mA,Port_IntDisable,
};
```

DIO:

```
    /* DIO Configured Port's ID  */
    #define DIO_CONFIG_LED1_PORT                (enuDioPort_t)PORTF
    #define DIO_CONFIG_LED2_PORT                (enuDioPort_t)PORTF
    #define DIO_CONFIG_LED3_PORT                (enuDioPort_t)PORTF
    #define DIO_CONFIG_SWITCH1_PORT             (enuDioPort_t)PORTF
    #define DIO_CONFIG_SWITCH2_PORT             (enuDioPort_t)PORTF

    /* DIO Configured Channel's ID */
    #define DIO_CONFIG_LED1_CHANNEL             (enuDioPin_t)PIN1
    #define DIO_CONFIG_LED2_CHANNEL             (enuDioPin_t)PIN2
    #define DIO_CONFIG_LED3_CHANNEL             (enuDioPin_t)PIN3
    #define DIO_CONFIG_SWITCH1_CHANNEL          (enuDioPin_t)PIN4
    #define DIO_CONFIG_SWITCH2_CHANNEL          (enuDioPin_t)PIN0
```

DIO Lcfg :

```
const strDioConfig_t ConfigList =
{
  DIO_CONFIG_LED1_PORT, DIO_CONFIG_LED1_CHANNEL,                  /* LED 1 @ PF1 */
  DIO_CONFIG_LED2_PORT, DIO_CONFIG_LED2_CHANNEL,                  /* LED 2 @ PF2 */
  DIO_CONFIG_LED3_PORT, DIO_CONFIG_LED3_CHANNEL,                  /* LED 3 @ PF3 */
  DIO_CONFIG_SWITCH1_PORT, DIO_CONFIG_SWITCH1_CHANNEL,            /* Switch 1 @ PF0 */
  DIO_CONFIG_SWITCH2_PORT, DIO_CONFIG_SWITCH2_CHANNEL             /* Switch 2 @ PF4 */
};
```

GPT Lcfg:

```
{
    TIMER0_16BIT, DISABLED,  UP, 16000000, GPT_CHANNEL_MODE_CONTINOUS, Gpt_Notification_0, 0,
    TIMER1_16BIT, DISABLED, UP, 16000000, GPT_CHANNEL_MODE_CONTINOUS, Gpt_Notification_1, 0,
    TIMER2_16BIT, DISABLED, UP, 16000000, GPT_CHANNEL_MODE_CONTINOUS, Gpt_Notification_2, 0,
    TIMER3_16BIT, DISABLED, UP, 16000000, GPT_CHANNEL_MODE_CONTINOUS, Gpt_Notification_3, 0,
    TIMER4_16BIT, DISABLED, UP, 16000000, GPT_CHANNEL_MODE_CONTINOUS, Gpt_Notification_4, 0,
    TIMER5_16BIT, DISABLED, UP, 16000000, GPT_CHANNEL_MODE_CONTINOUS, Gpt_Notification_5, 0,
    TIMER0_32BIT, DISABLED, UP, 16000000, GPT_CHANNEL_MODE_CONTINOUS, Gpt_Notification_6, 0,
    TIMER1_32BIT, DISABLED, UP, 16000000, GPT_CHANNEL_MODE_CONTINOUS, Gpt_Notification_7, 0,
    TIMER2_32BIT, DISABLED, UP, 16000000, GPT_CHANNEL_MODE_CONTINOUS, Gpt_Notification_8, 0,
    TIMER3_32BIT, DISABLED, UP, 16000000, GPT_CHANNEL_MODE_CONTINOUS, Gpt_Notification_9, 0,
    TIMER4_32BIT, DISABLED, UP, 16000000, GPT_CHANNEL_MODE_CONTINOUS, Gpt_Notification_10, 0,
    TIMER5_32BIT, DISABLED, UP, 16000000, GPT_CHANNEL_MODE_CONTINOUS, Gpt_Notification_11, 0,
};
```