Moving Car Project V2.0

Team #2

- Amr El-Abd
- Ahmed Atef
- Anas Mahmoud
- Khaled Mustafa

Table of Contents: -

Project introduction:

This project involves designing a system for a four-diving wheel robot to move in a rectangular shape. By leveraging the unique capabilities of this robot configuration, we aim to develop a precise and efficient control system. Understanding the mechanics and kinematics of the robot will be essential for achieving accurate movement. Through intelligent control algorithms, we will coordinate the speed, direction, and synchronization of the four wheels to ensure smooth navigation along the predefined rectangular path. This project holds great potential for enhancing mobility and expanding the applications of four-diving wheel robots.

Project description :-

1. Hardware Requirements

1. Use the **TivaC launch pad**
2. **Four** motors (**M1**, **M2**, **M3**, **M4**)
3. **One** button to start (**PB1**)
4. **One** button for stop (**PB2**)
5. **Four** LEDs (**LED1**, **LED2**, **LED3**, **LED4**)

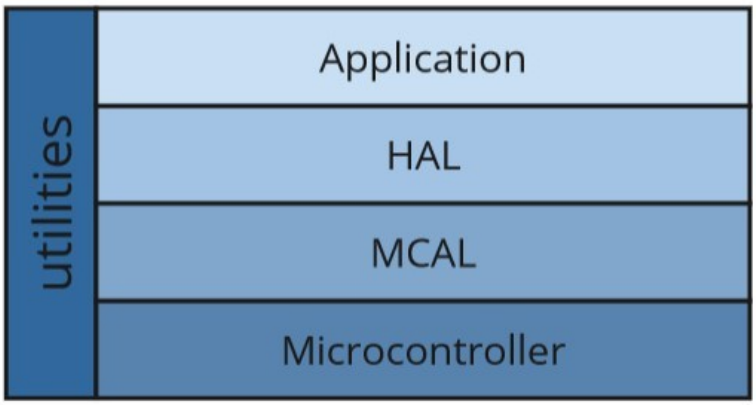
2. Software Requirements

The car **starts initially** from **0 speed**

- When **PB1** is **pressed**, the car will **move forward after 1 second**
- The car will move forward to **create the longest side of the rectangle for 3 seconds with 50% of its maximum speed**
- After finishing the first longest side the car will **stop for 0.5 seconds**, **rotate 90 degrees to the right**, and **stop for 0.5 second**
- The car will move to **create the short side** of the rectangle at **30% of its speed for 2 seconds**
- After finishing the shortest side, the car will stop for **0.5 seconds**, **rotate 90 degrees to the right**, and **stop for 0.5 second**
- Steps **3 to 6** will be **repeated infinitely** until you press the **stop button (PB2)**
- **PB2** acts as a **sudden break**, and it has the highest priority
- **LEDs Operations**

1. **LED1:** On means moving forward on the long side
2. **LED2:** On means moving forward on the short side
3. **LED3:** On means stop
4. **LED4:** On means Rotating

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.

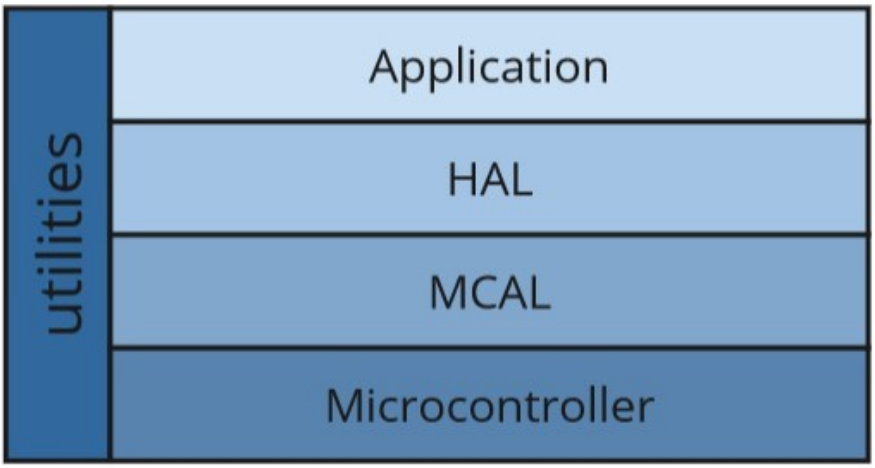
HAL Layer: This layer provides an abstraction for external devices connected to the microcontroller. The HAL layer provides interface to access external devices and hides the implementation details from the application layer.

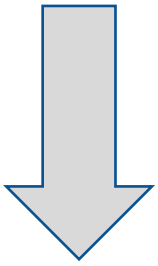
MCAL Layer (Microcontroller Abstraction Layer): This layer provides an abstraction for the microcontroller hardware. It includes low-level drivers for peripherals. It hides the hardware details and provides a uniform interface to the upper layers.

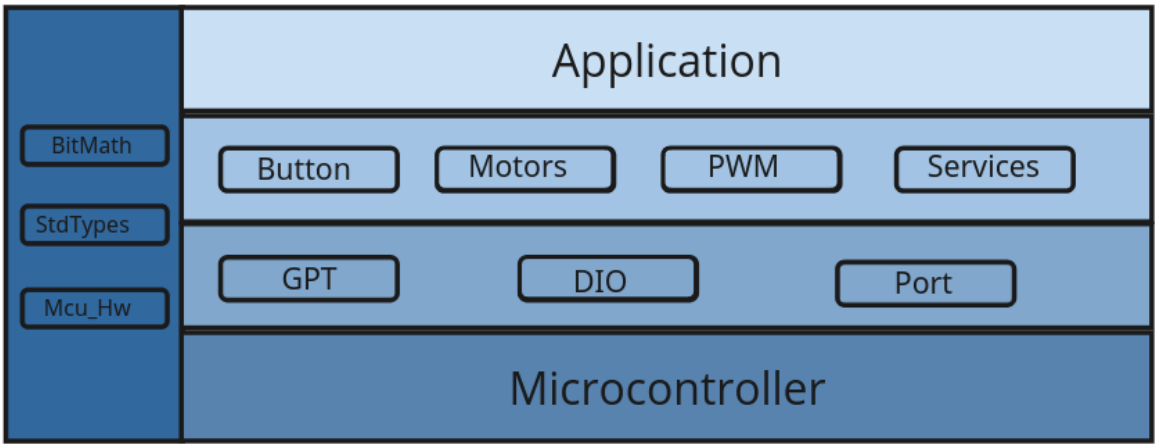
Utilities Layer: the utilities layer includes memory mapping, standard types, and utils.h. Memory mapping involves defining the memory layout and addresses for different components. Standard types provide a set of predefined data types that ensure consistency and portability across different platforms. 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:-


utilities
Application
HAL
MCAL
Microcontroller




BitMath
StdTypes
Mcu_Hw
Application
Button
Motors
PWM
Services
GPT
DIO
Port
Microcontroller

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);

Motor Driver

- Initializes the motor pin.
- Moves the motors in the forward direction.
- Moves the motors in the backward direction.
- Stops the movements of the motors.
- Rotate the motors to the right.

APIs

en_MOTOR_Status_t MOTOR_Init

en_MOTOR_Status_t MOTOR_Forward

en_MOTOR_Status_t MOTOR_Backward

en_MOTOR_Status_t MOTOR_RotateRight

en_MOTOR_Status_t MOTOR_RotateLeft

en_MOTOR_Status_t MOTOR_Stop

PWM Driver

Description: this driver provides an interface for controlling the PWM module of the TIVA C MCU.

APIs

enu_ErrorReturn Blink_Stop

enu_ErrorReturn Pwm_Init

enu_ErrorReturn Pwm_Start

enu_ErrorReturn Pwm_Stop

Service Driver

Description: This driver provides an interface for controlling and interfacing with the timer module of the TIVA C MCU.

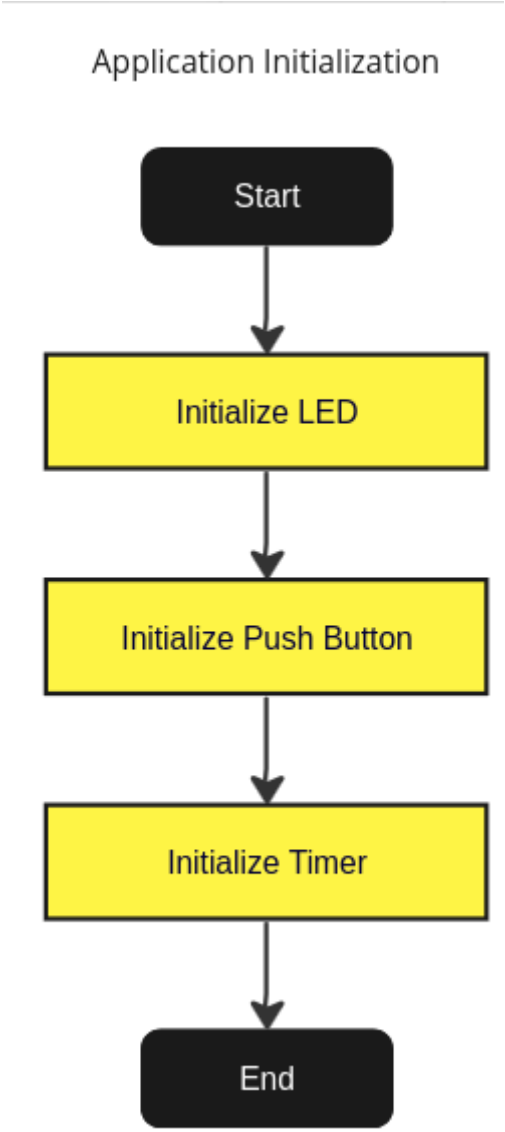
APIs

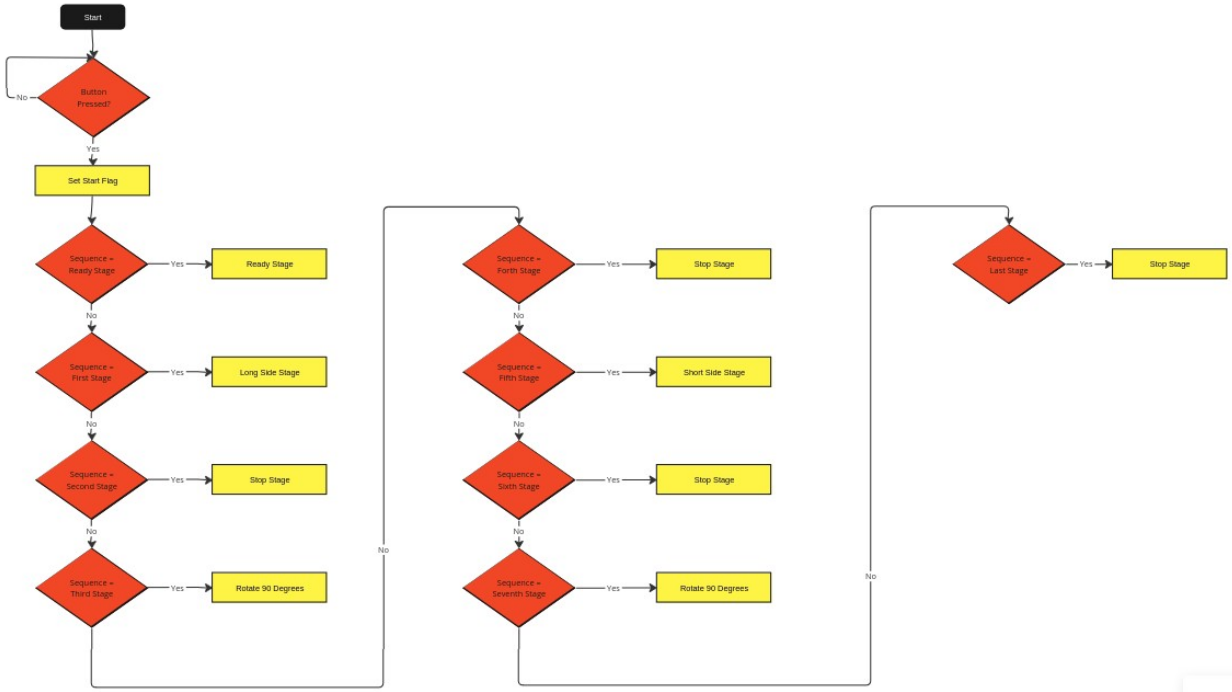
enu_ErrorReturn Service_TimerInit

enu_ErrorReturn Service_TimerStart

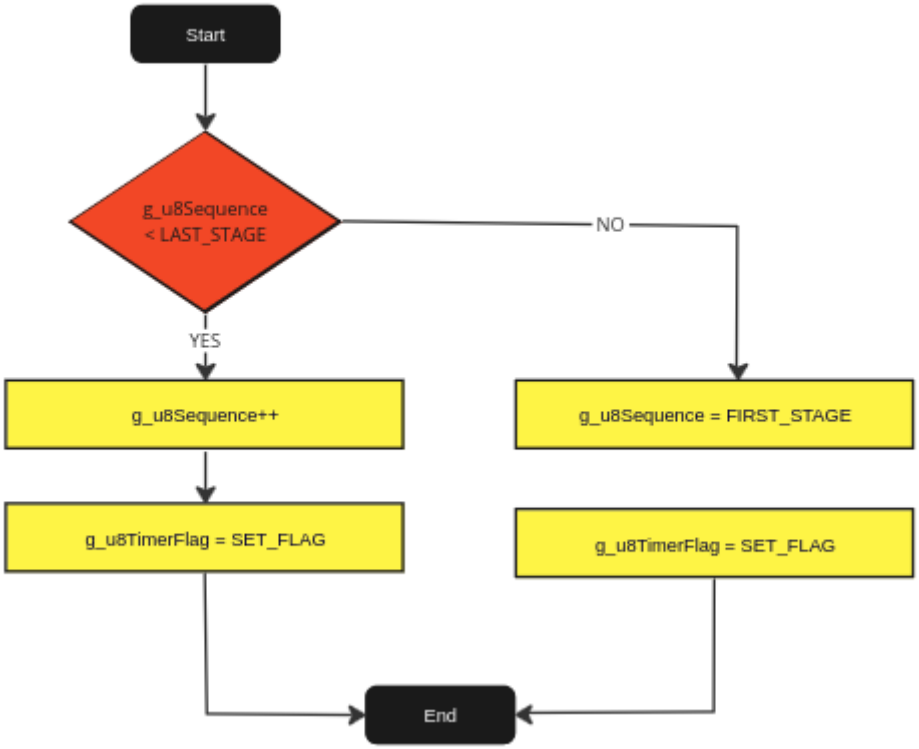
Flowcharts for Functions:

Application

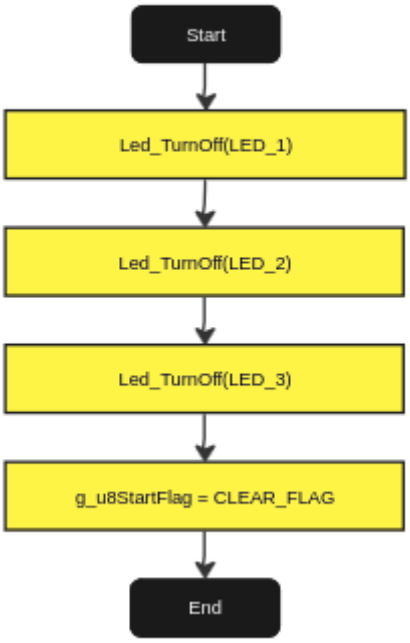




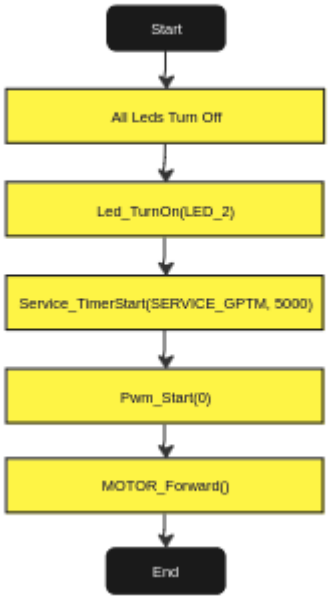
void App_SequenceChange (void)



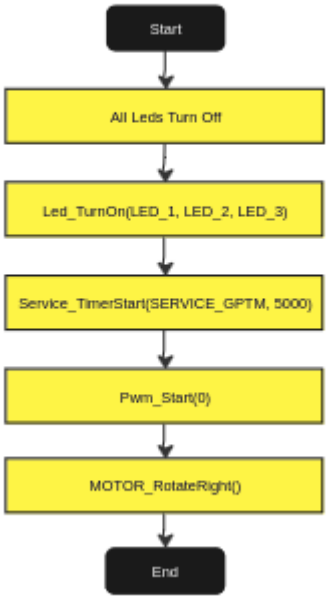
void App_ButtonInterrupt (void)



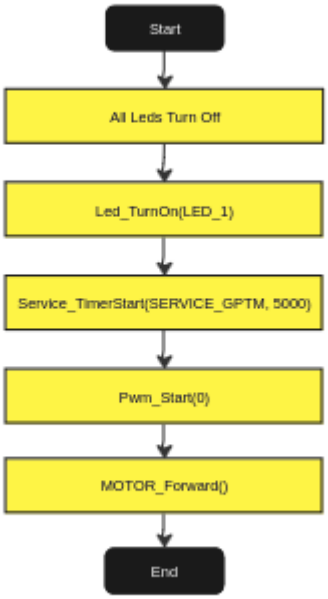
void App_ShortSideStage()



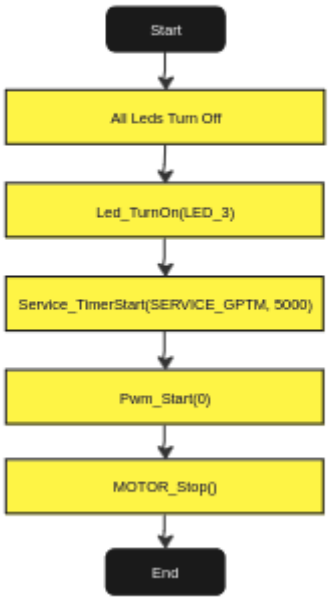
void App_Rotate90degreeCalculation (void)



void App_LongSideStage(void)

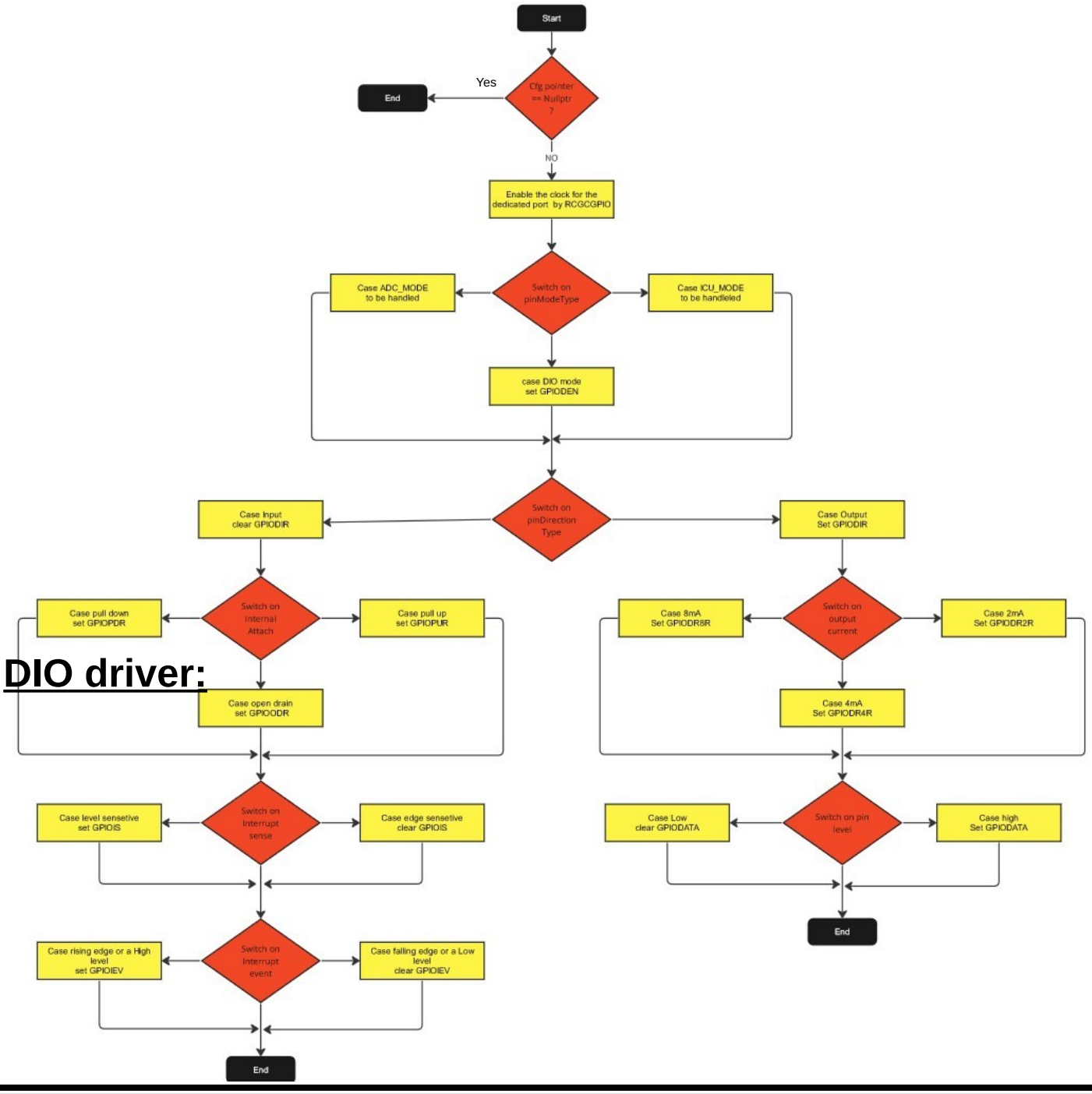


void App_CarStopStage (void)

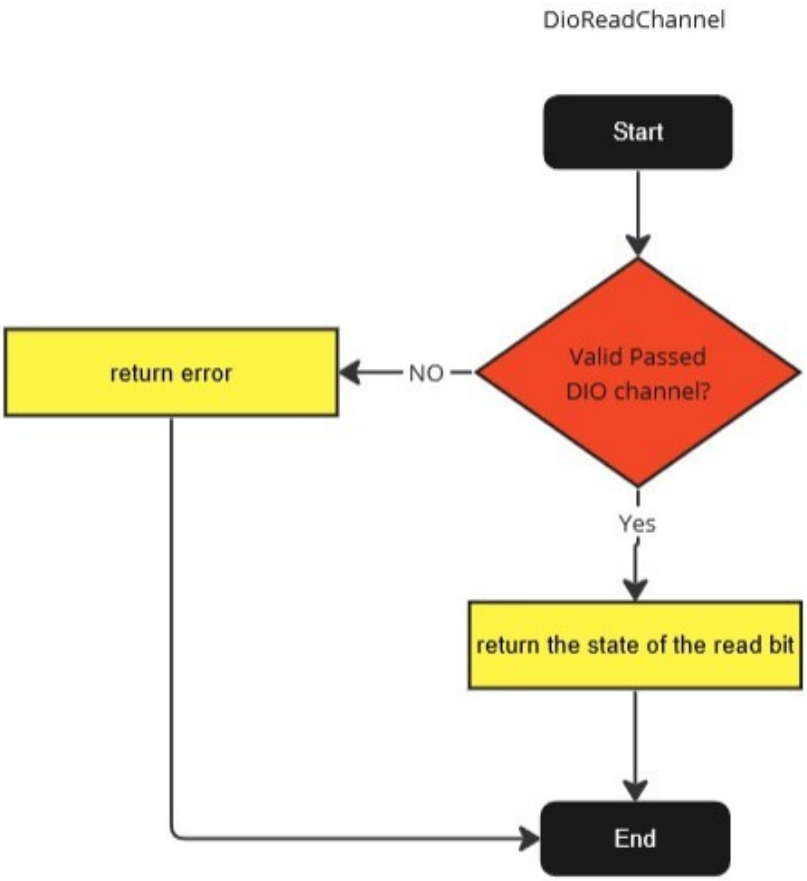


Port driver:

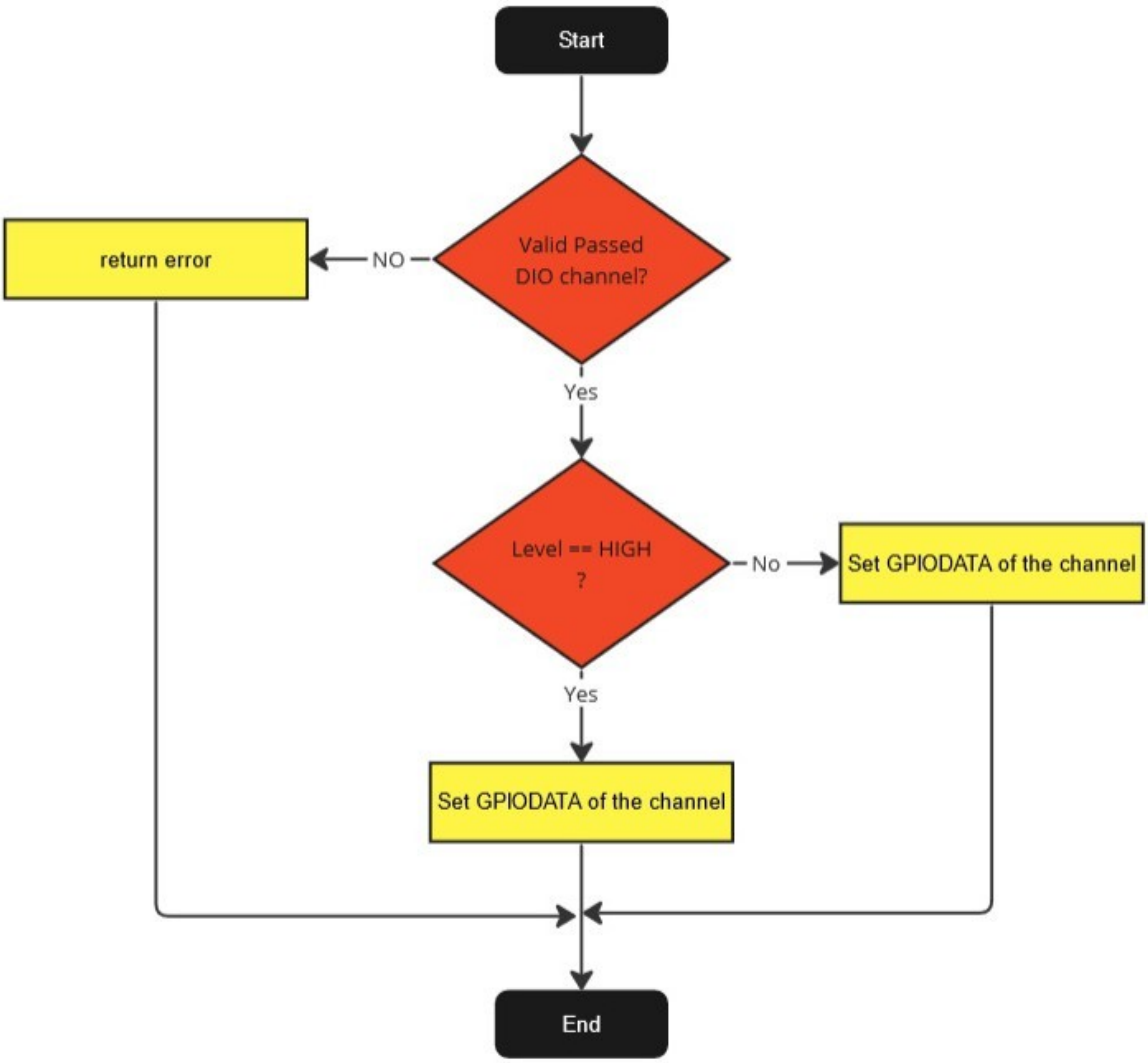
Port init :



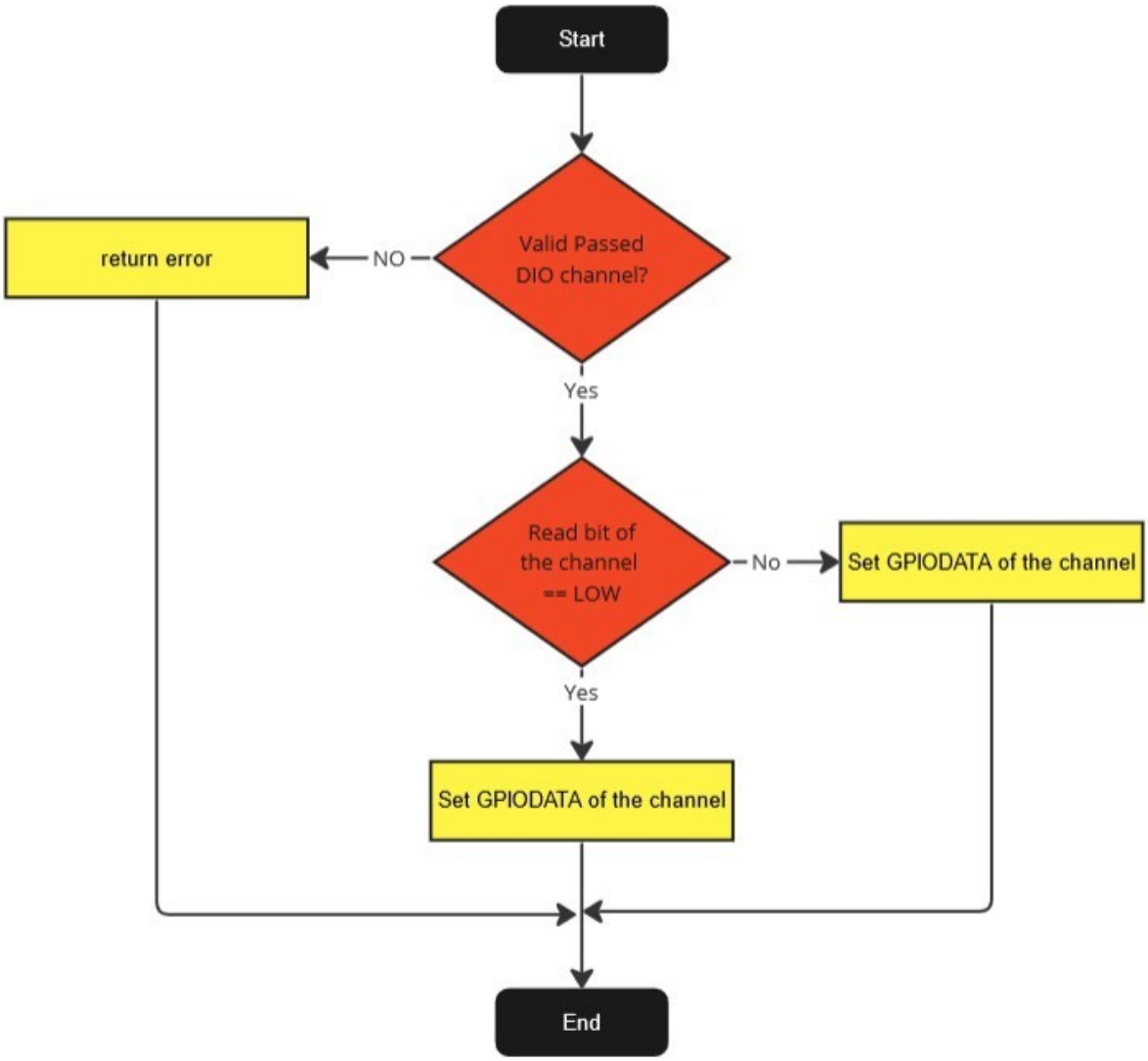
DIO driver:


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

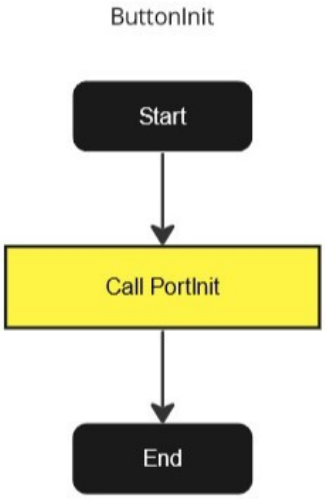
DioWriteChannel



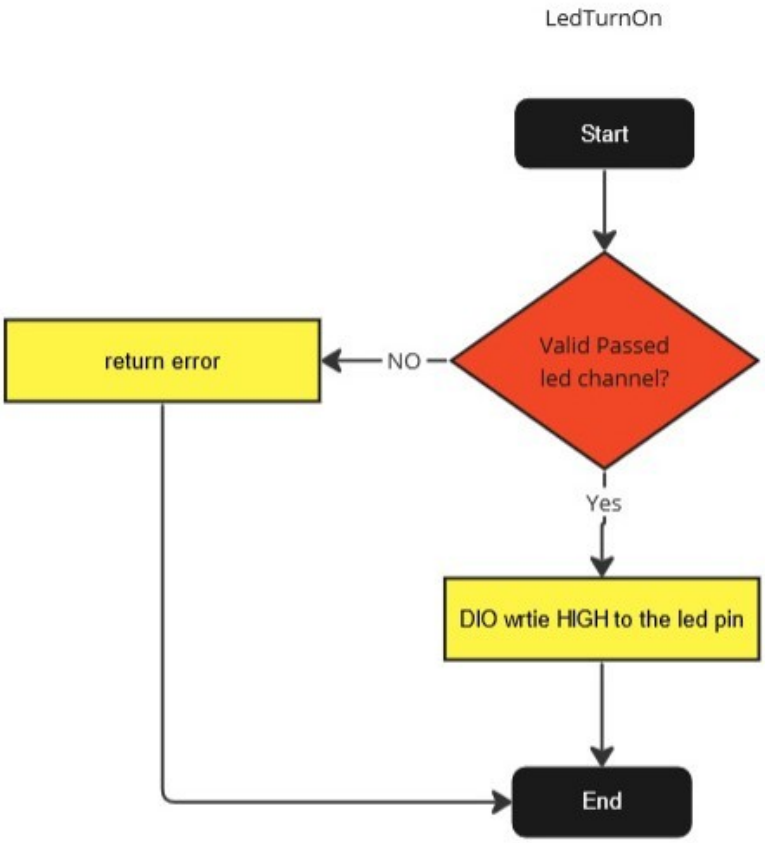
DioToggleChannel

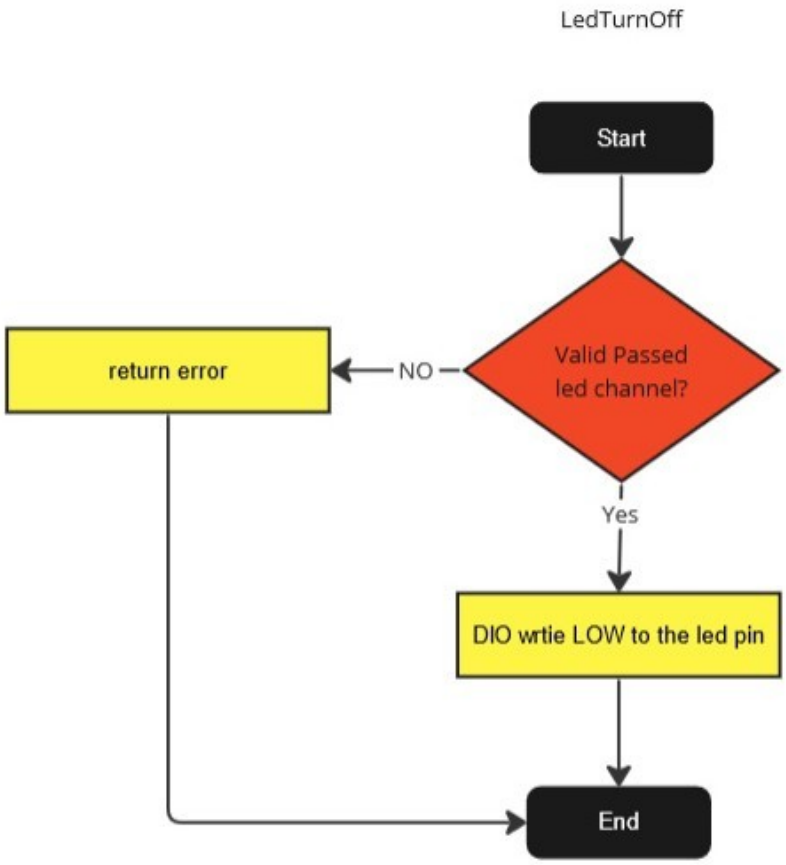


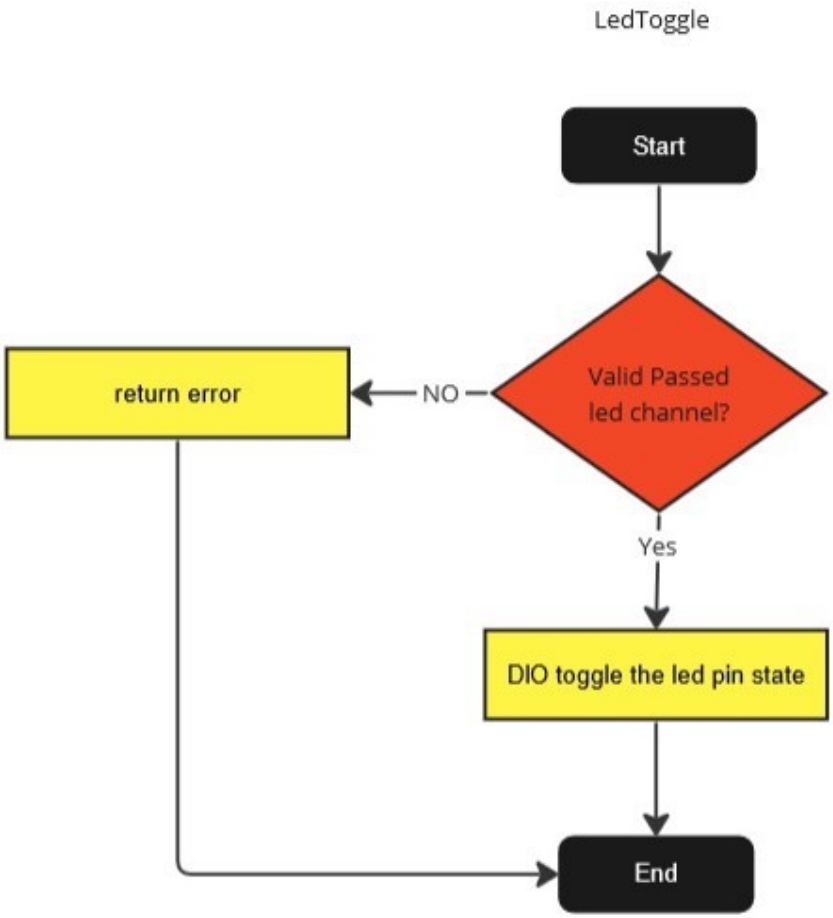
Button driver



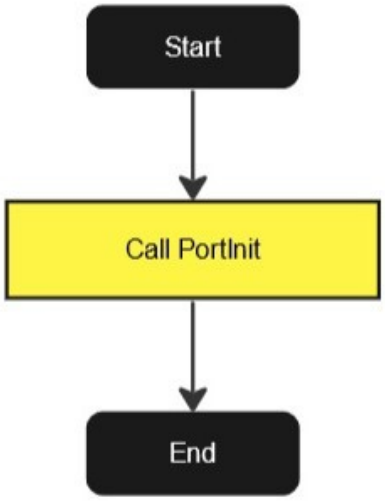
LED driver:


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

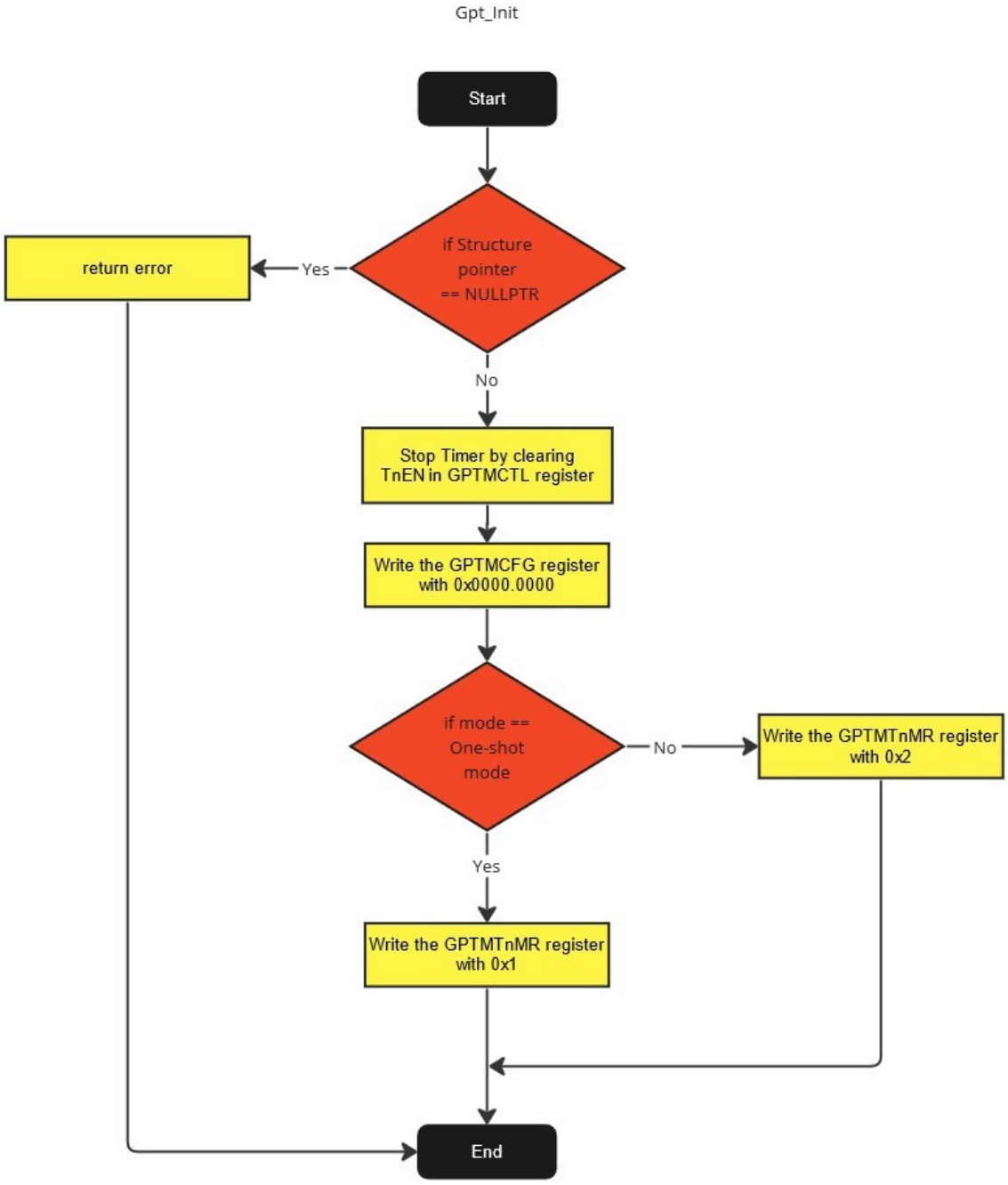

LedTurnOff
Start
Valid Passed
led channel?
NO
return error
Yes
DIO wrtie LOW to the led pin
End


LedToggle
Start
Valid Passed
led channel?
NO
return error
Yes
DIO toggle the led pin state
End

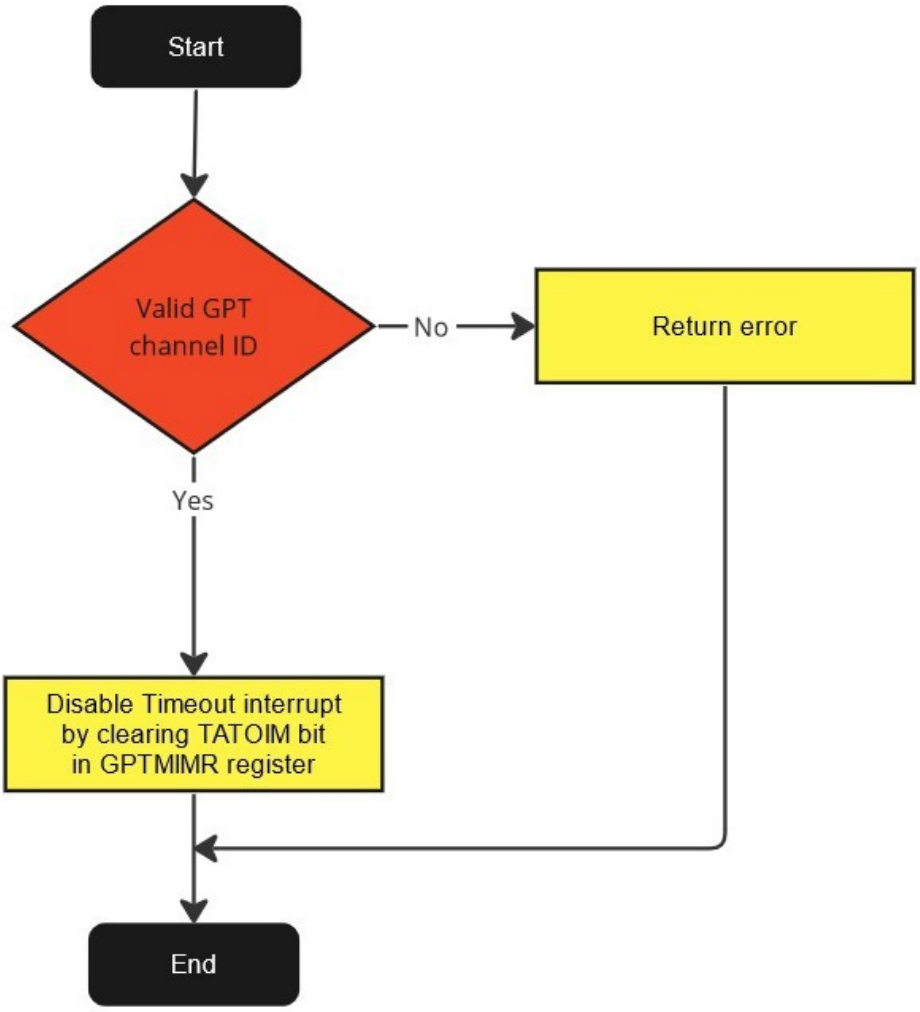
LedInit



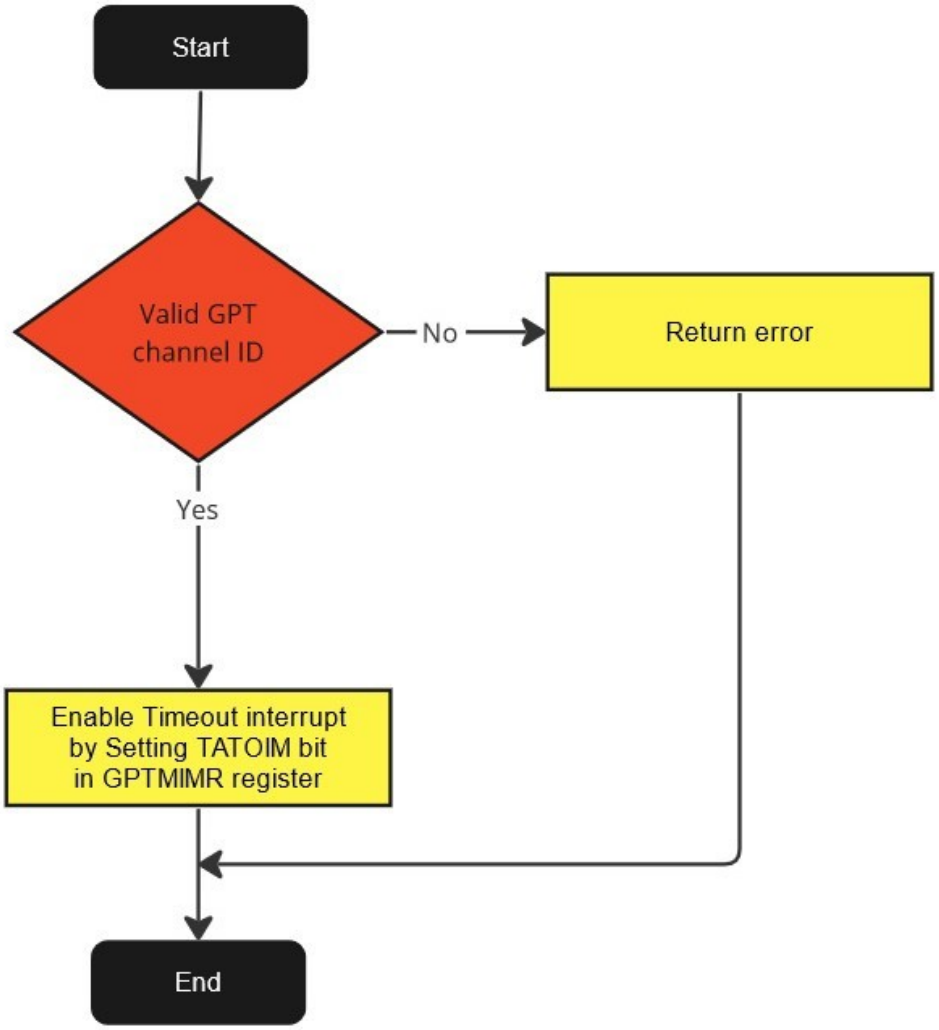
GPT Functions:-



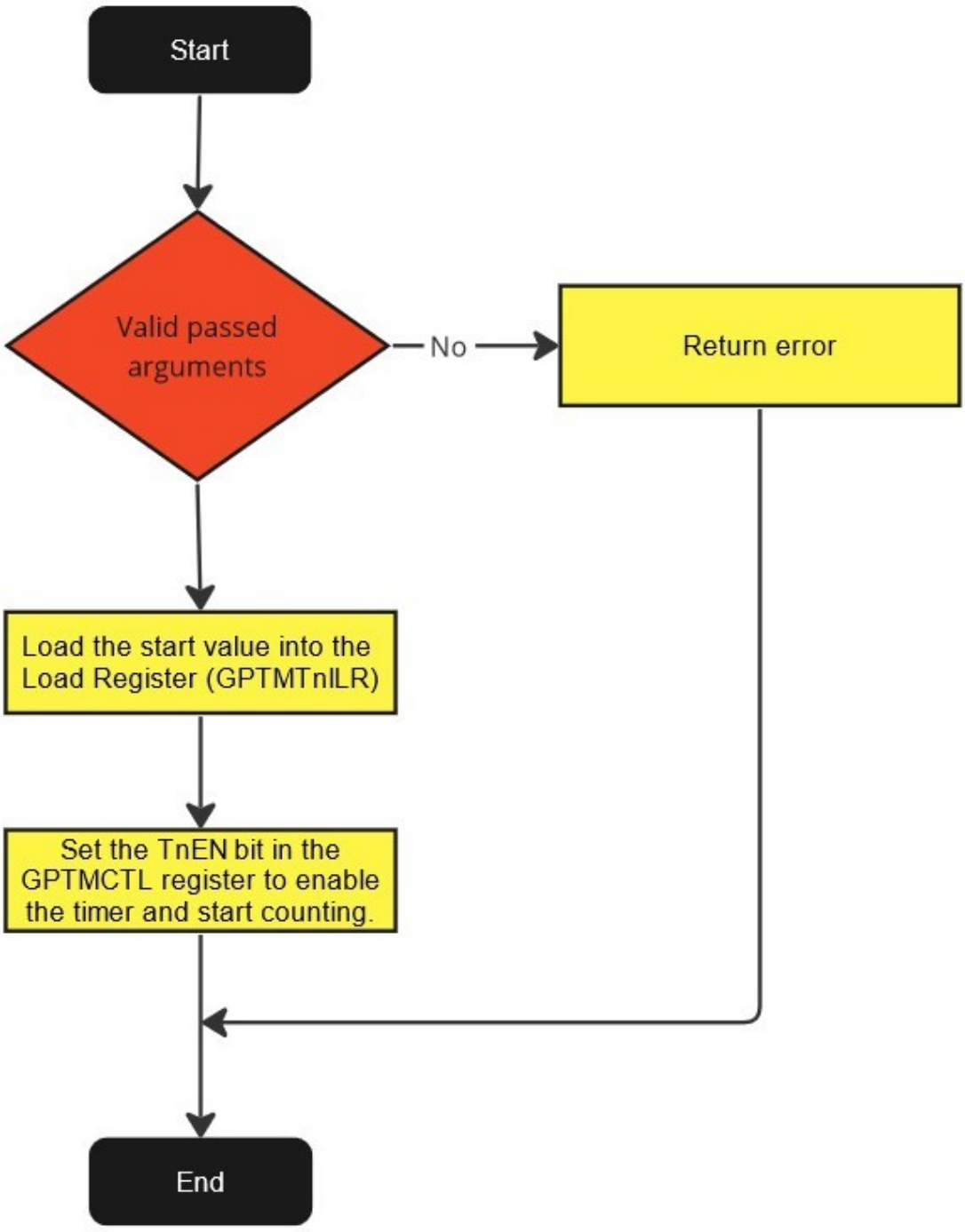
Gpt_DisableNotification



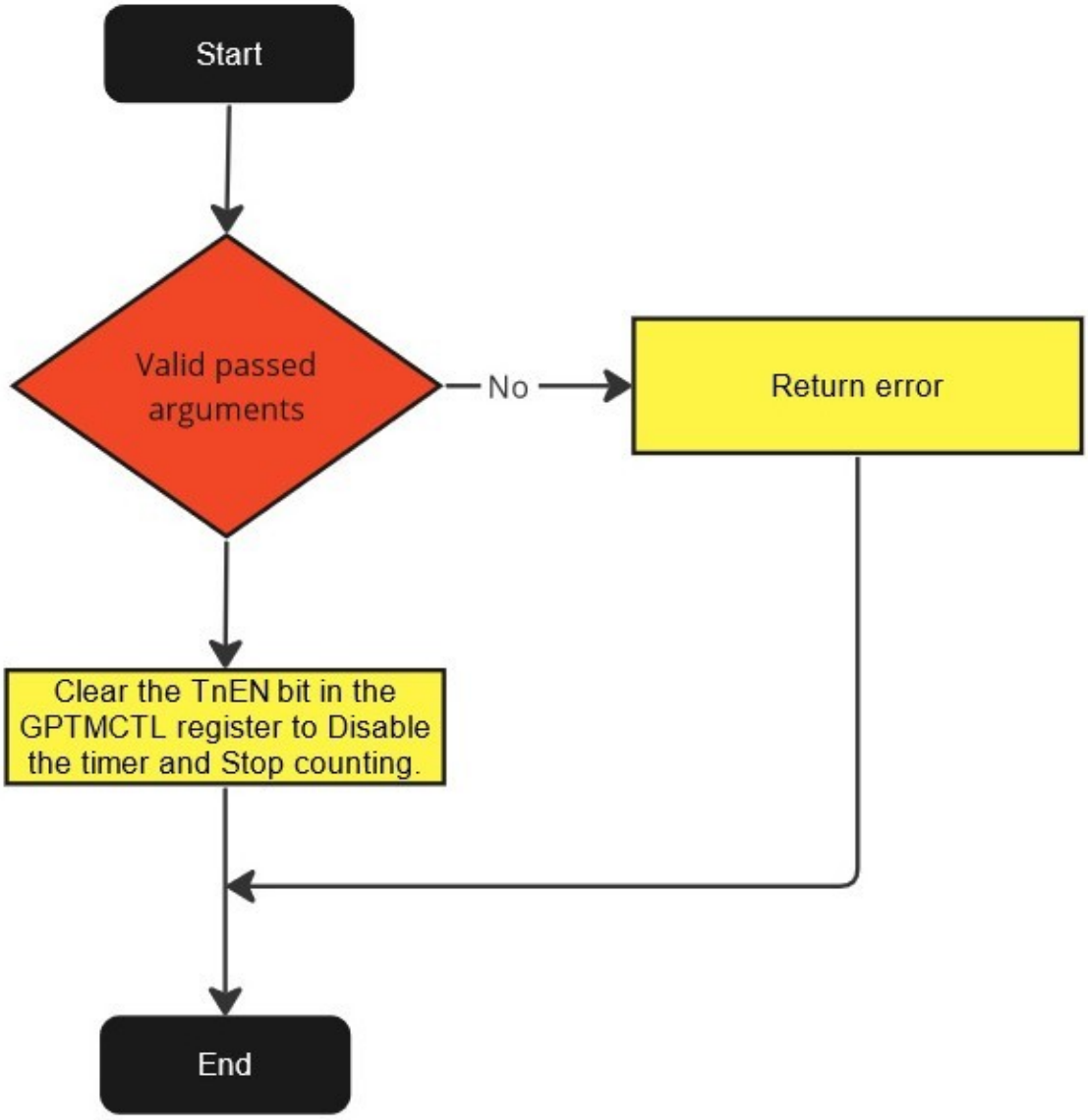
Gpt_EnableNotification



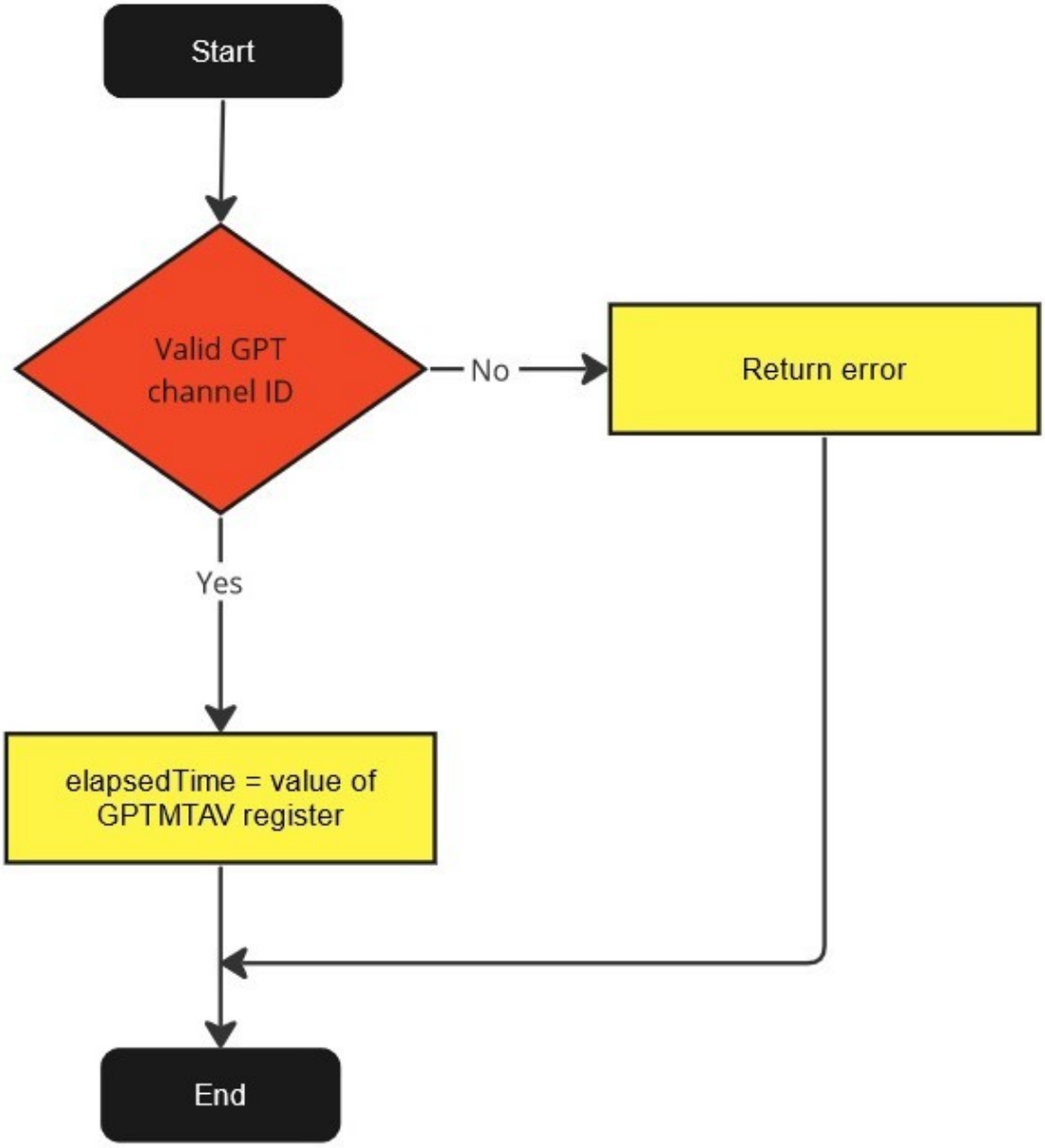
Gpt_StartTimer



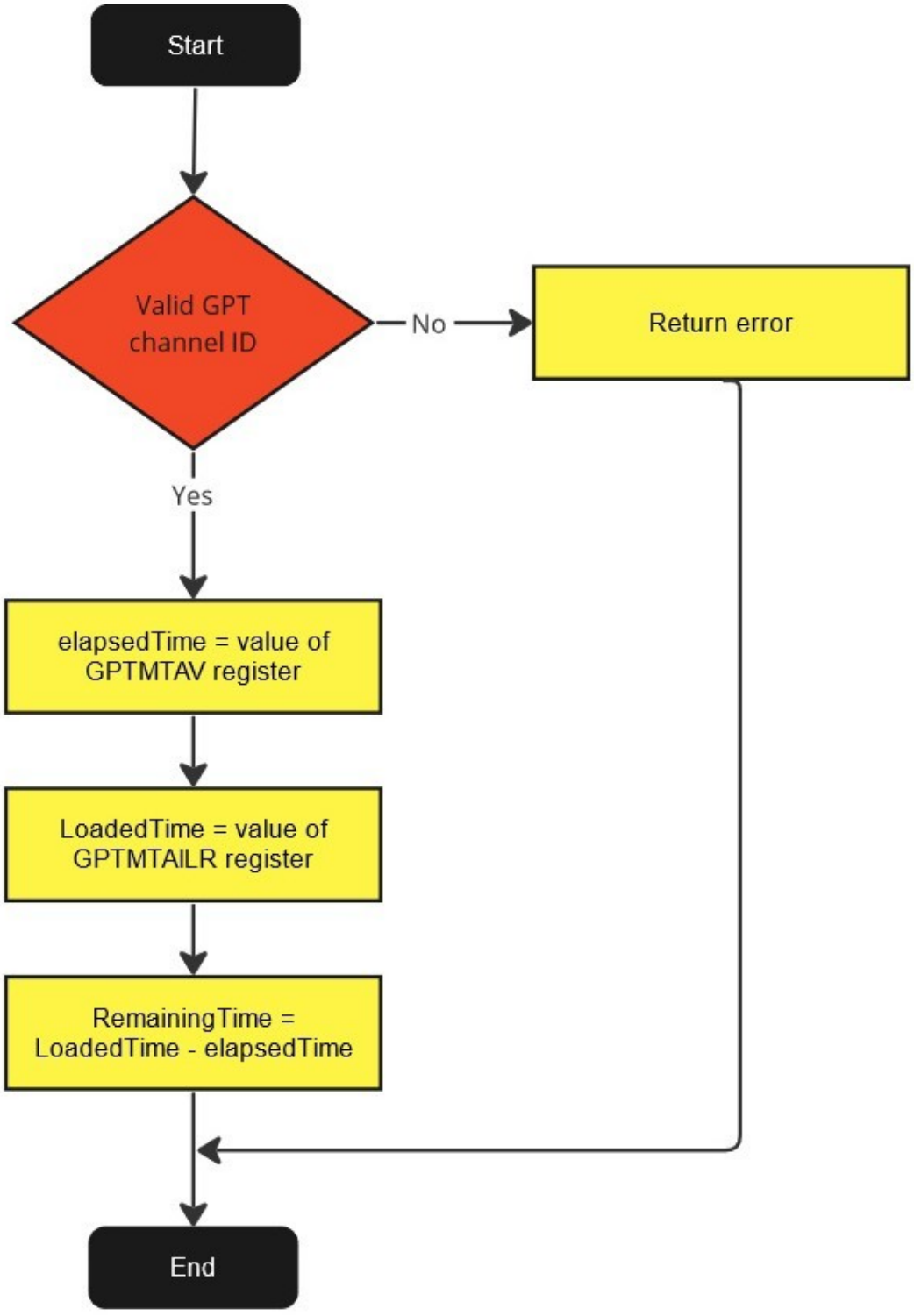
Gpt_StopTimer



Gpt_GetTimeElapsed

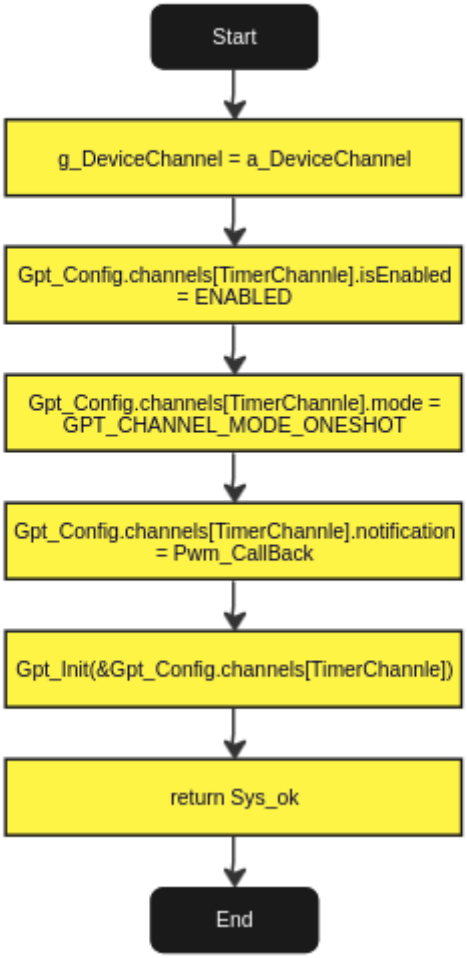


Gpt_GetTimeElapsed

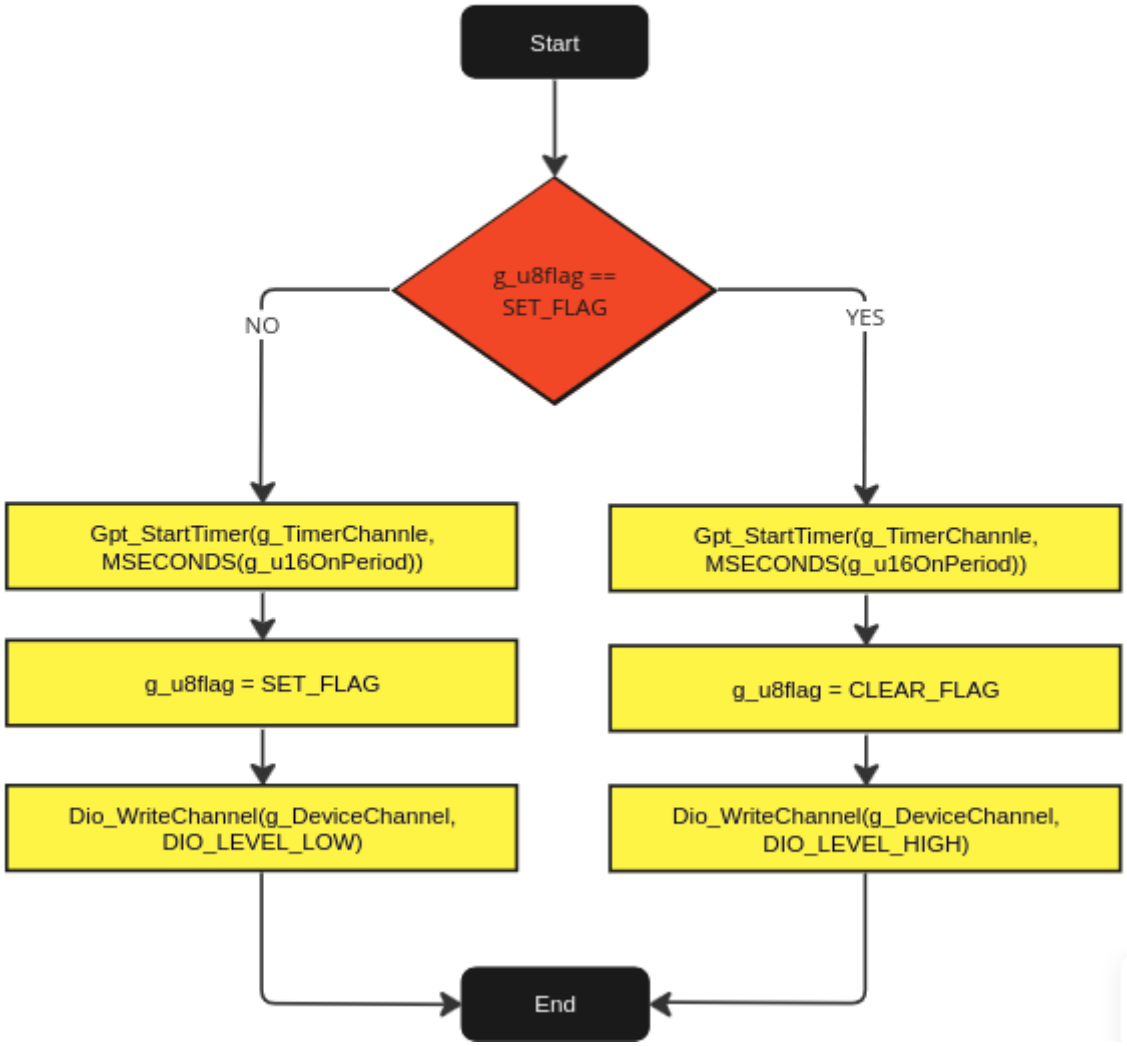


PWM Module

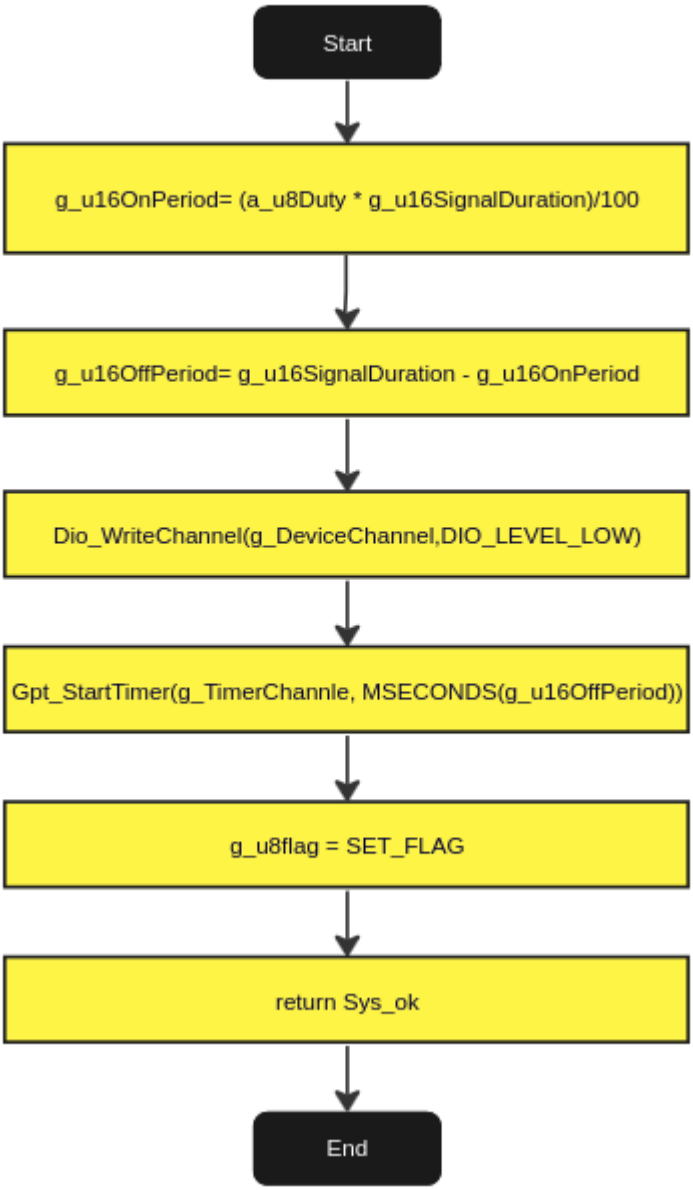
enu_ErrorReturn Pwm_Init (Service_TimerChannelType TimerChannle,
Service_DeviceChannel a_DeviceChannel)



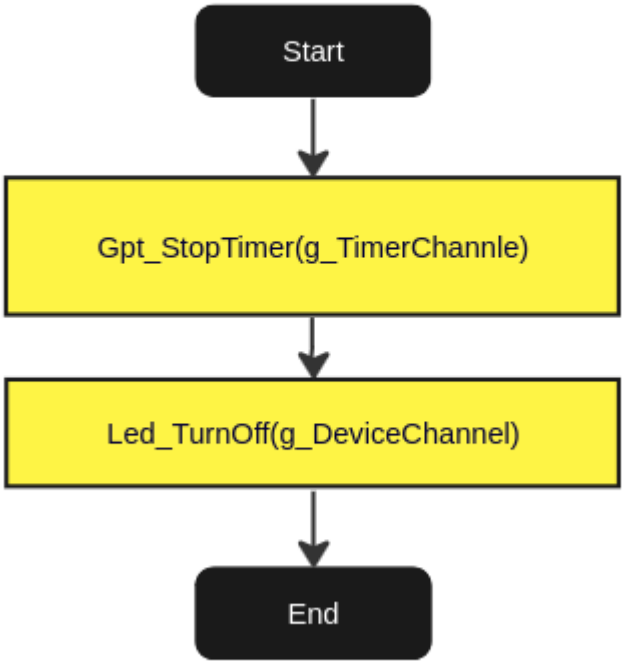
void Pwm_CallBack(void)



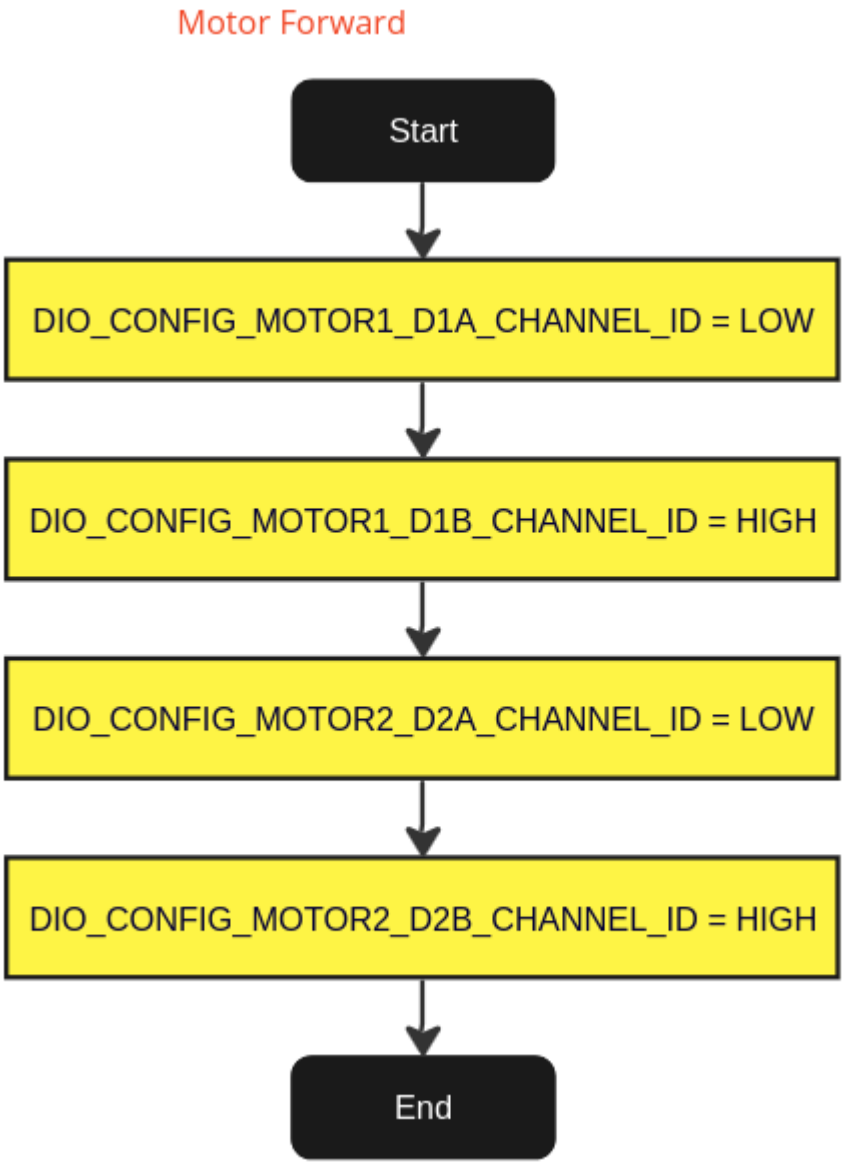
enu_ErrorReturn Pwm_Start(uint8 a_u8Duty)



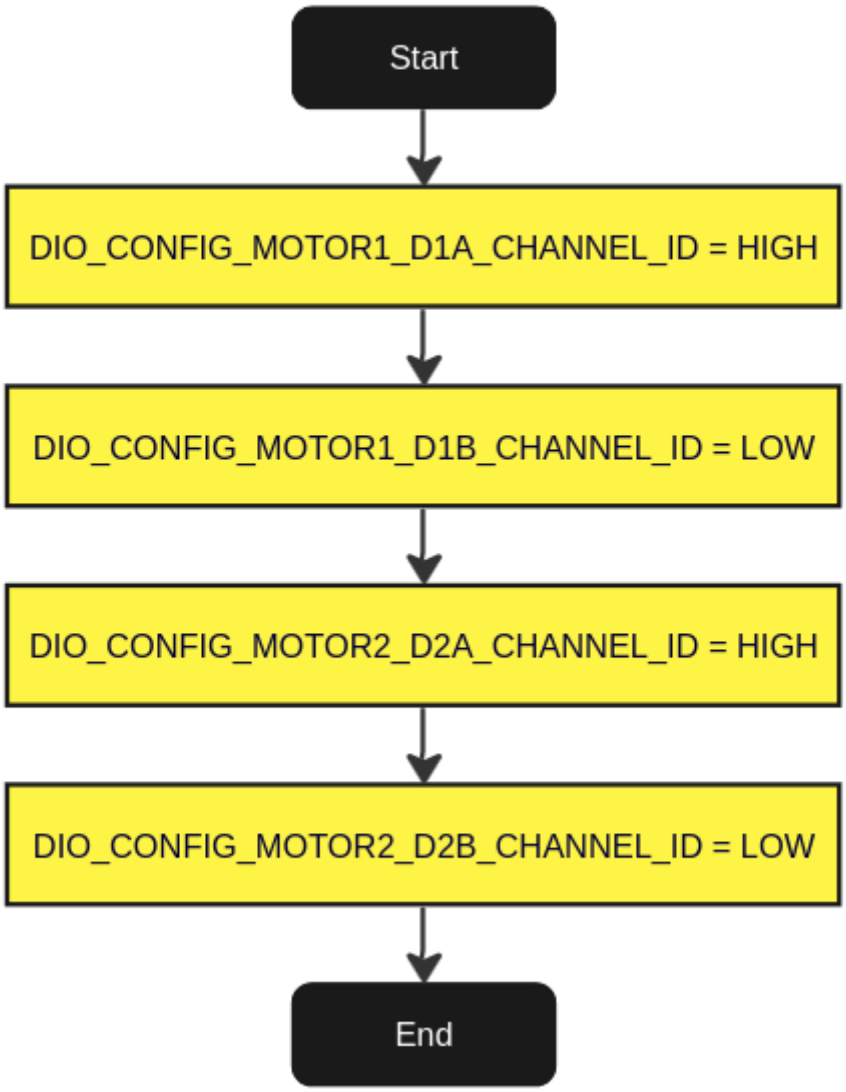
enu_ErrorReturn Pwm_Stop(void)



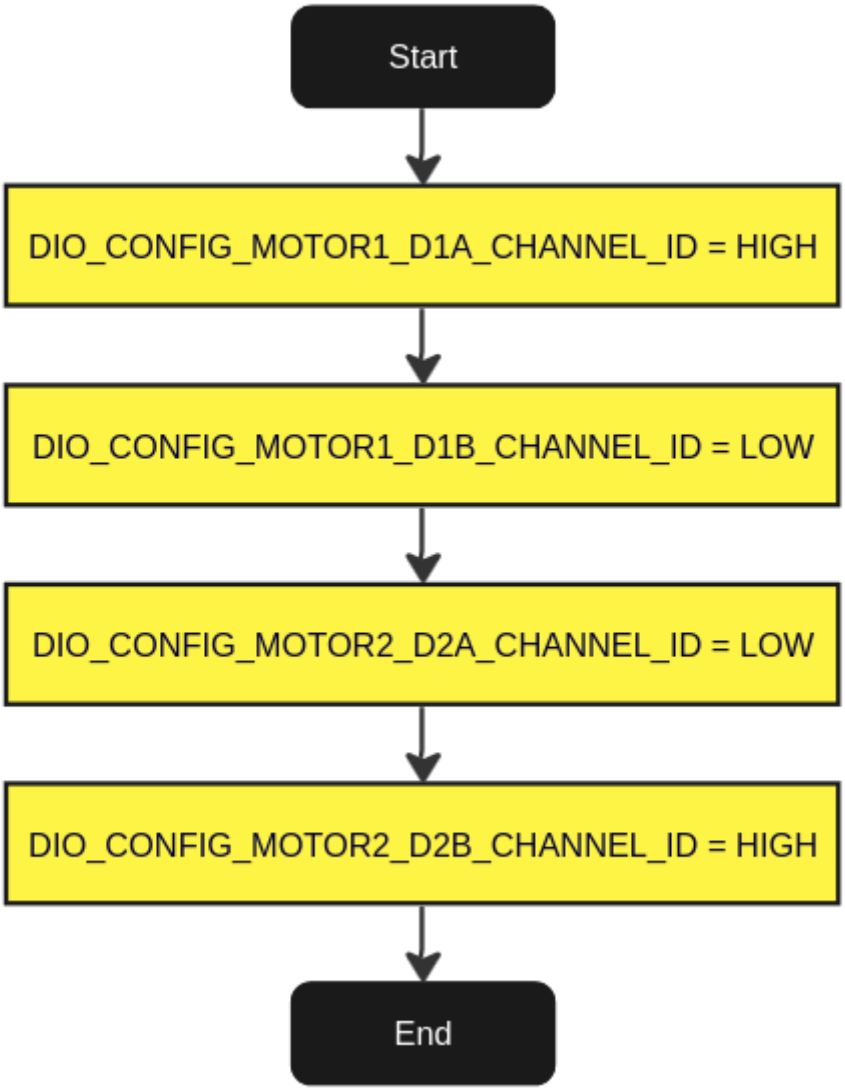
Motor Module



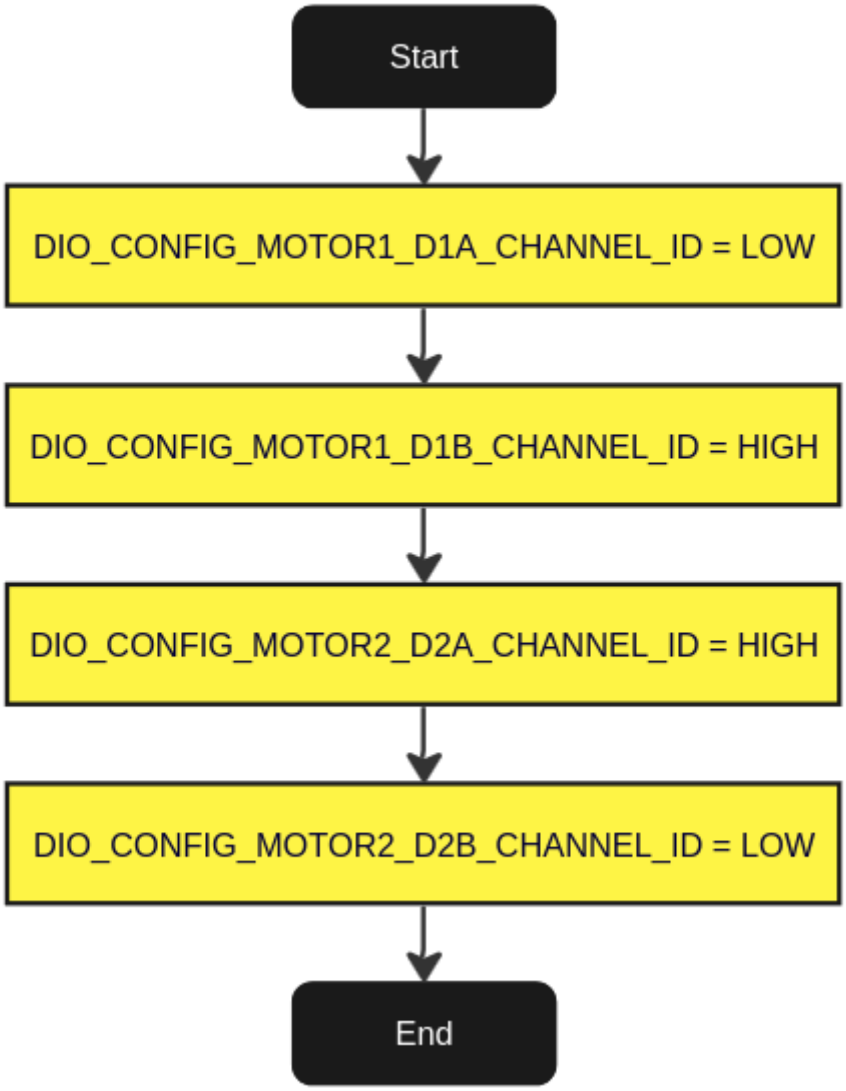
Motor Backward



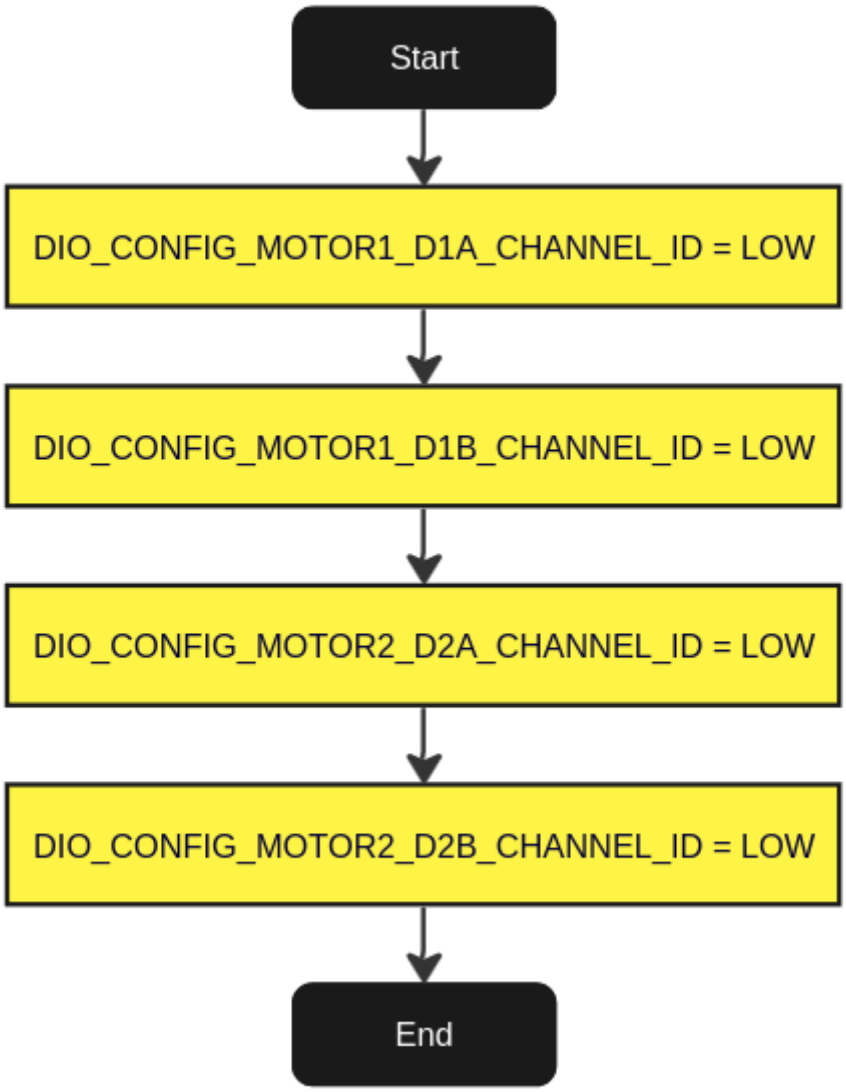
Motor Rotate Right



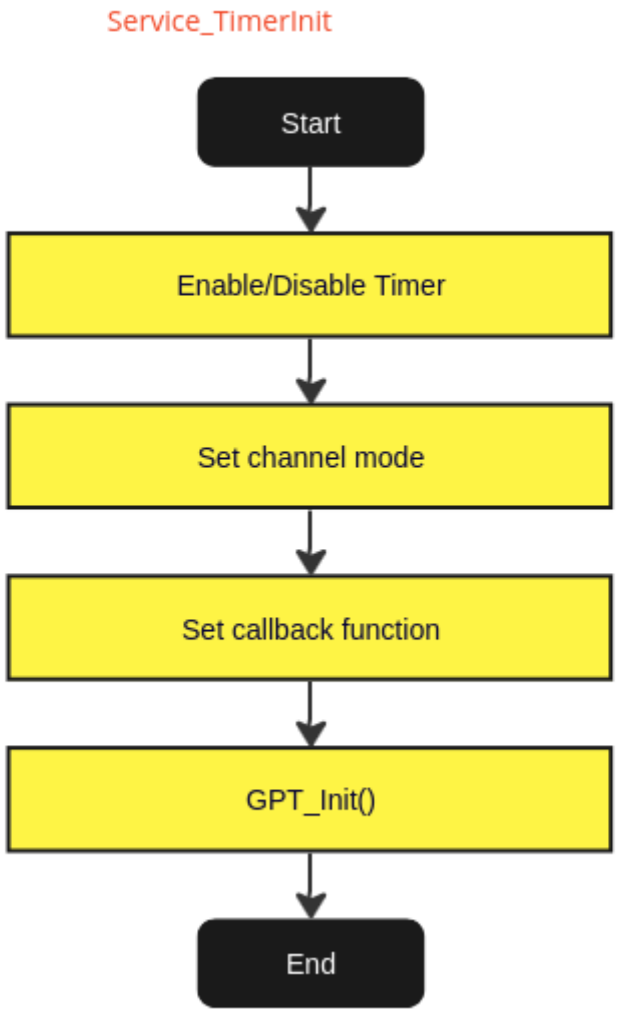
Motor Rotate Left

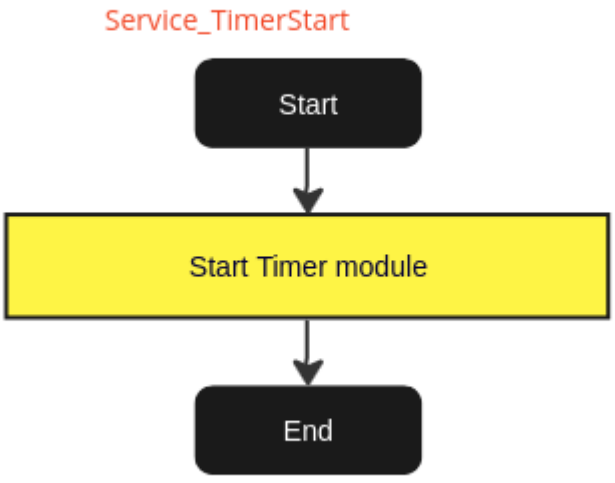


Motor Stop


Start
DIO_CONFIG_MOTOR1_D1A_CHANNEL_ID = LOW
DIO_CONFIG_MOTOR1_D1B_CHANNEL_ID = LOW
DIO_CONFIG_MOTOR2_D2A_CHANNEL_ID = LOW
DIO_CONFIG_MOTOR2_D2B_CHANNEL_ID = LOW
End

Service Driver





Precompiling and linking configurations :-

Port Lcfg:

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

    /********************************** PORTB ***********************************/
    PORTB, PIN0, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    PORTB, PIN1, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    PORTB, PIN2, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    PORTB, PIN3, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    PORTB, PIN4, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    PORTB, PIN5, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    PORTB, PIN6, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    PORTB, PIN7, CHANNEL_ENABLED, OUTPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_DOWN,
DRIVE_2mA,

    /********************************** PORTC ***********************************/
    PORTC, PIN0, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    // Will be ignored, because it is reserved for [JTAG/SWD]
    PORTC, PIN1, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    // Will be ignored, because it is reserved for [JTAG/SWD]
    PORTC, PIN2, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    // Will be ignored, because it is reserved for [JTAG/SWD]
    PORTC, PIN3, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    // Will be ignored, because it is reserved for [JTAG/SWD]
    PORTC, PIN4, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    PORTC, PIN5, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    PORTC, PIN6, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    PORTC, PIN7, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,

    /********************************** PORTD ***********************************/
    PORTD, PIN0, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    PORTD, PIN1, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    PORTD, PIN2, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    PORTD, PIN3, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    PORTD, PIN4, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    PORTD, PIN5, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    PORTD, PIN6, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    PORTD, PIN7, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,

    /********************************** PORTE ***********************************/
    PORTE, PIN0, CHANNEL_DISABLED, INPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_UP, DRIVE_2mA,
    PORTE, PIN1, CHANNEL_ENABLED, OUTPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_DOWN,
DRIVE_2mA,
    PORTE, PIN2, CHANNEL_ENABLED, OUTPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_DOWN,
DRIVE_2mA,
    PORTE, PIN3, CHANNEL_ENABLED, OUTPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_DOWN,
DRIVE_2mA,
```

```
    PORTE, PIN4, CHANNEL_ENABLED, OUTPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_DOWN,
DRIVE_2mA,
    PORTE, PIN5, CHANNEL_ENABLED, OUTPUT, PIN_LEVEL_LOW, DIO_MODE, PULL_DOWN,
DRIVE_2mA,

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

DIO:

```
 /* Number of the configured Dio Channels */
#define NUM_CONFIGURED_CHANNELS                    (11U)  //should be 12 for EN
/*----------------------------------INDEX-----------------------------------*/
/* Channel Index in the array of structures in Dio_Lcfg.c */
#define DIO_CONFIG_LED1_CHANNEL_ID                      (uint8)0x00
#define DIO_CONFIG_LED2_CHANNEL_ID                      (uint8)0x01
#define DIO_CONFIG_LED3_CHANNEL_ID                      (uint8)0x02
#define DIO_CONFIG_LED4_CHANNEL_ID                      (uint8)0x03

#define DIO_CONFIG_SWITCH1_CHANNEL_ID              (uint8)0x04
#define DIO_CONFIG_SWITCH2_CHANNEL_ID              (uint8)0x05

#define DIO_CONFIG_MOTOR1_D1A_CHANNEL_ID                    (uint8)0x06
#define DIO_CONFIG_MOTOR1_D1B_CHANNEL_ID                      (uint8)0x07
#define DIO_CONFIG_MOTOR2_D2A_CHANNEL_ID                    (uint8)0x08
#define DIO_CONFIG_MOTOR2_D2B_CHANNEL_ID                    (uint8)0x09

#define DIO_CONFIG_MOTOR1_EN_CHANNEL_ID                       (uint8)0x0A
//#define DIO_CONFIG_MOTOR2_EN_CHANNEL_ID                     (uint8)0x0B
/*---------------------------------PORT_ID----------------------------------*/
/* DIO Configured Port's ID  */
#define DIO_CONFIG_LED1_PORT                       (Dio_PortType)PORTF
#define DIO_CONFIG_LED2_PORT                       (Dio_PortType)PORTF
#define DIO_CONFIG_LED3_PORT                       (Dio_PortType)PORTF
#define DIO_CONFIG_LED4_PORT                       (Dio_PortType)PORTE

#define DIO_CONFIG_SWITCH1_PORT                    (Dio_PortType)PORTF
#define DIO_CONFIG_SWITCH2_PORT                    (Dio_PortType)PORTF


/* Motor Ports ID */
#define DIO_CONFIG_MOTOR1_D1A_PORT                      (Dio_PortType)PORTE
#define DIO_CONFIG_MOTOR1_D1B_PORT                        (Dio_PortType)PORTE
#define DIO_CONFIG_MOTOR2_D2A_PORT                      (Dio_PortType)PORTE
#define DIO_CONFIG_MOTOR2_D2B_PORT                      (Dio_PortType)PORTE

#define DIO_CONFIG_MOTOR1_EN_PORT                         (Dio_PortType)PORTB
//#define DIO_CONFIG_MOTOR2_EN_PORT                       (Dio_PortType)PORTF

/*-----------------------------Channel's ID-----------------------------*/

/* DIO Configured Channel's ID */
#define DIO_CONFIG_LED1_CHANNEL                    (Dio_PinType)PIN1
#define DIO_CONFIG_LED2_CHANNEL                    (Dio_PinType)PIN2
#define DIO_CONFIG_LED3_CHANNEL                    (Dio_PinType)PIN3
```

```
#define DIO_CONFIG_LED4_CHANNEL                         (Dio_PinType)PIN1


#define DIO_CONFIG_SWITCH1_CHANNEL                      (Dio_PinType)PIN4
#define DIO_CONFIG_SWITCH2_CHANNEL                      (Dio_PinType)PIN0

/* Motor Channel ID */
#define DIO_CONFIG_MOTOR1_D1A_CHANNEL                           PIN2
#define DIO_CONFIG_MOTOR1_D1B_CHANNEL                           PIN3
#define DIO_CONFIG_MOTOR2_D2A_CHANNEL                           PIN4
#define DIO_CONFIG_MOTOR2_D2B_CHANNEL                           PIN5

#define DIO_CONFIG_MOTOR1_EN_CHANNEL                            PIN7
//#define DIO_CONFIG_MOTOR2_EN_CHANNEL                            PIN0
/*-------------------------------------------------------------------------*/
/
*******************************************************************************
**********************************
 *  GLOBAL DATA PROTOTYPES

*******************************************************************************
**********************************/
/* Structure to gather the configured channels to be used and manipulated easily */
typedef struct
{
      Dio_ChannelConfigType channels[NUM_CONFIGURED_CHANNELS];
} Dio_ConfigType;

/
*******************************************************************************
**********************************
 *  GLOBAL DATA TYPES AND STRUCTURES

*******************************************************************************
**********************************/
extern const Dio_ConfigType configList;
```

DIO Lcfg

```
/
*******************************************************************************************
***************************
 *  GLOBAL DATA

*******************************************************************************************
***************************/
const Dio_ConfigType 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_LED4_PORT, DIO_CONFIG_LED4_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 */

 DIO_CONFIG_MOTOR1_D1A_PORT,DIO_CONFIG_MOTOR1_D1A_CHANNEL,
 DIO_CONFIG_MOTOR1_D1B_PORT,DIO_CONFIG_MOTOR1_D1B_CHANNEL,
```

```
 DIO_CONFIG_MOTOR2_D2A_PORT,DIO_CONFIG_MOTOR2_D2A_CHANNEL,
 DIO_CONFIG_MOTOR2_D2B_PORT,DIO_CONFIG_MOTOR2_D2B_CHANNEL,

 DIO_CONFIG_MOTOR1_EN_PORT,DIO_CONFIG_MOTOR1_EN_CHANNEL,
// DIO_CONFIG_MOTOR2_EN_PORT ,DIO_CONFIG_MOTOR2_EN_CHANNEL,
};
```

GPT Lcfg:

```
    /
    ********************************************************************************************
    ******************************
     *  GLOBAL CONSTANT MACROS

    ********************************************************************************************
    ******************************/
    #define GPT_PREDEF_TIMER_1US_16BIT                                  (0U)
    #define GPT_PREDEF_TIMER_1US_24BIT                                  (0U)
    #define GPT_PREDEF_TIMER_1US_32BIT                                  (0U)
    #define GPT_PREDEF_TIMER_100US_32BIT                                (0U)

    /
    ********************************************************************************************
    ******************************
     *  GLOBAL DATA TYPES AND STRUCTURES

    ********************************************************************************************
    ******************************/
    extern int Timer0_Counter;
    extern Gpt_ConfigType Gpt_Config;
```