

# DESIGN DOCUMENT

RGB LED CONTROL

V2.0 DESIGN

Team2:

-Amr El-Abd

-Ahmed Aatef

## **Table of Contents: -**

<b>Subject</b>	<b>Page</b>
• Project introduction.....	3
• Project description.....	3
• High Level Design	
1. Layered architecture.....	4
2. Modules Descriptions.....	5
3. Drivers' documentation.....	6
• Low Level Design	
1. Flowchart for each function in each module.....	10
2. Pre-compiling and linking configurations .....	22

# **Project introduction:**

The project aims to utilize the TivaC board, SW1 button, and RGB LED to create an interactive system. Initially, the RGB LED is turned off. When the SW1 button is pressed, the system responds in a sequential manner. On the first press, the Red LED illuminates for a duration of 1 second. On the second press, the Green LED turns on for 1 second. The third press activates the Blue LED for 1 second. On the fourth press, all LEDs are simultaneously lit for 1 second. The fifth press disables all LEDs, and the sixth press restarts the cycle. This project provides an engaging user interface that allows for repetitive interaction and exploration of different LED colors.

## **Project description :-**

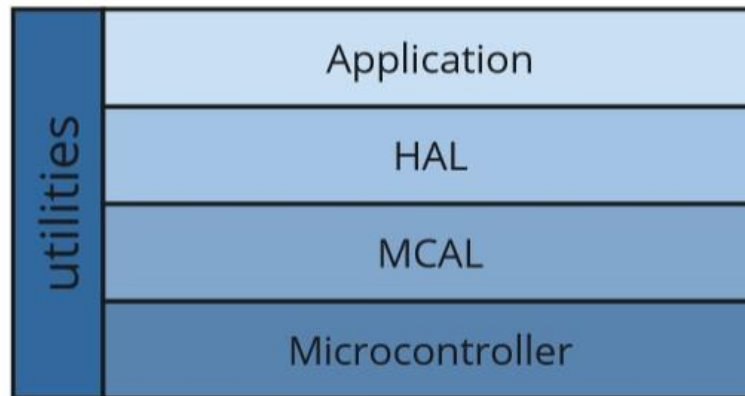
### **1. Hardware Requirements**

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. Pressing SW1:
  1. After the first press, the Red led is on **for 1 second only**.
  2. After the second press, the Green Led is on **for 1 second only**.
  3. After the third press, the Blue led is on **for 1 second only**.
  4. After the fourth press, all LEDs are on **for 1 second only**.
  5. After the fifth press, should disable all LEDs.
  6. After the sixth press, repeat steps from 1 to 6.

## 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.

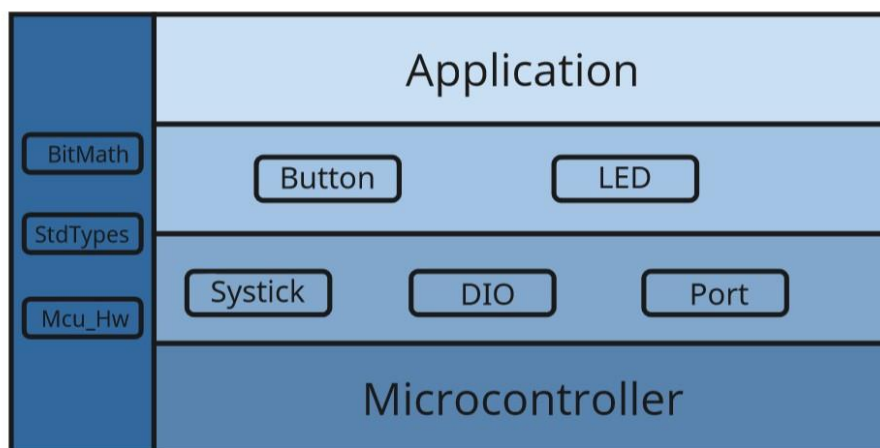
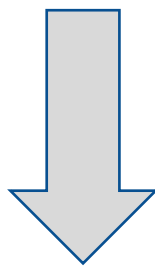
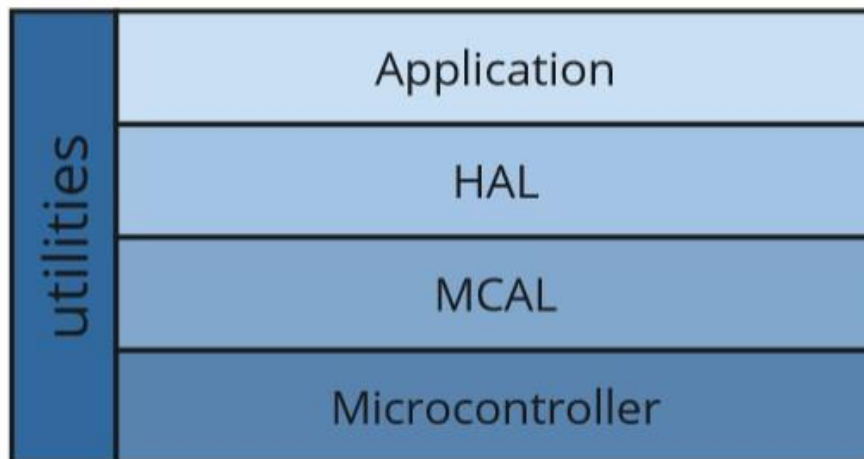
**HAL Layer:** This layer provides an abstraction for external devices connected to the microcontroller. The HAL layer provides interface to access external devices and hides the implementation details from the application layer.

**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 pin

### **APIs:**

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

## **DIO driver:**

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

### **APIs:**

```
enumDioLevel_t DioReadChannel(DioChannel_t ChannelId);
```

```
void DioWriteChannel(DioChannel_t ChannelId, enumDioLevel_t Level);
```

```
enumDioLevel_t DioToggleChannel(DioChannel_t ChannelId);
```

## **LED driver:**

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

### **APIs:**

```
void LedInit(void);
```

```
void LedTurnOn(LedChannel_t LedChannel);
```

```
void LedTurnOff(LedChannel_t LedChannel);
```

```
void LedToggle(LedChannel_t LedChannel);
```

## **Button driver:**

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

### **APIs:**

```
void ButtonInit(void);
```

```
enumButtonState_t ButtonGetState(ButtonChannel_t ButtonChannel,  
enumButtonAttach_t ButtonAttach);
```

## **Systick driver:**

Description: The SysTick driver provides a high-level interface and functionality to configure and utilize the SysTick timer for system timing and periodic interrupt purposes.

### **APIs:**

```
enumErrorStatus_t SysTick_Init (void);
```

```
enumErrorStatus_t SysTick_SetIntervalSingle(uint32 u32Ticks, void(*Ptrf)(void));
```

```
enumErrorStatus_t SysTick_SetIntervalPeriodic(uint32 u32Ticks, void(*Ptrf)(void));
```

```
enumErrorStatus_t SysTick_StopInterval(void);
```

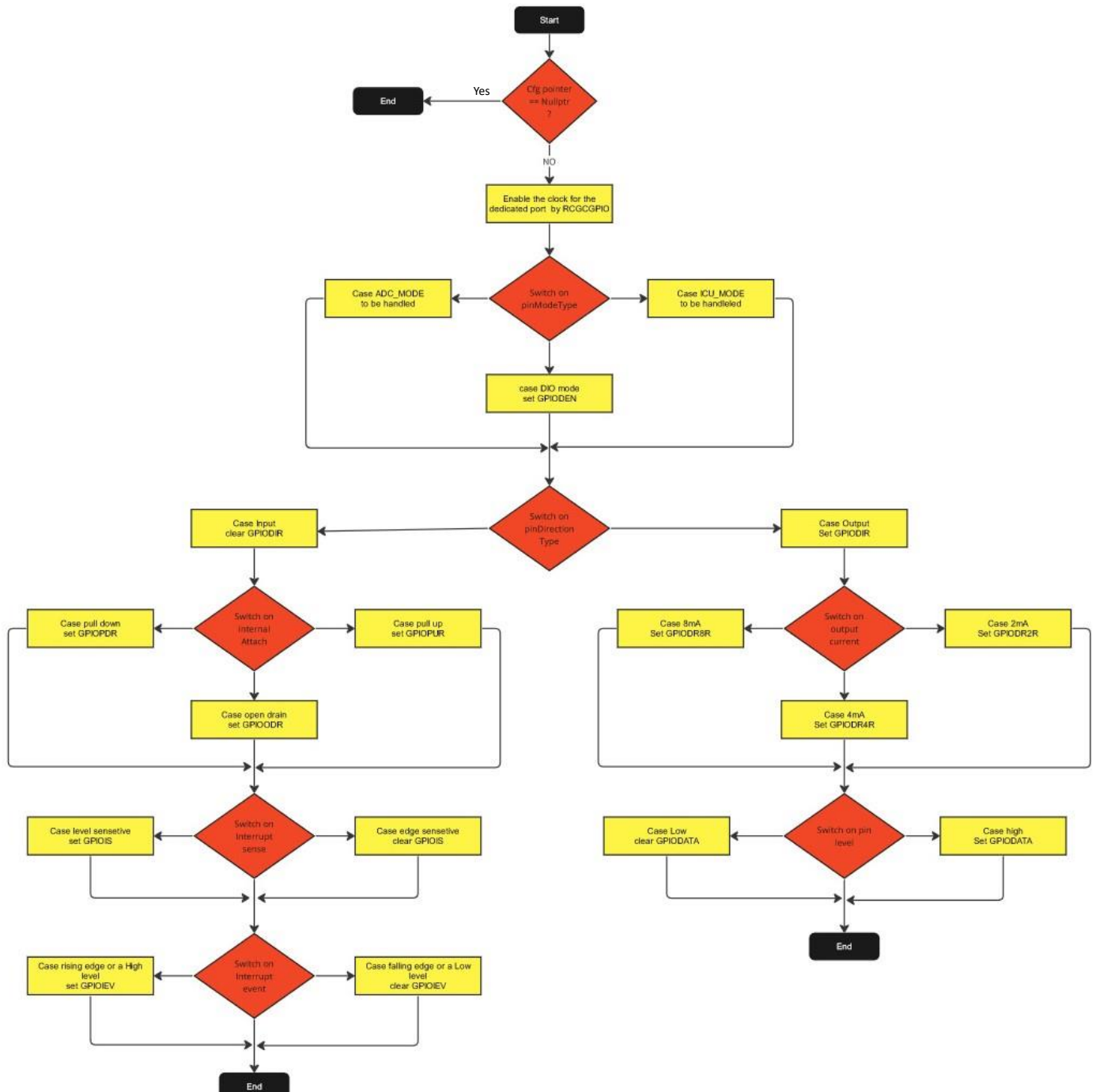
```
enumErrorStatus_t SysTick_GetElapsedTime(uint32* u32ElapsedTime);
```

```
enumErrorStatus_t SysTick_GetRemainingTime(uint32* u32RemainingTime);
```

## Flowcharts for Functions:

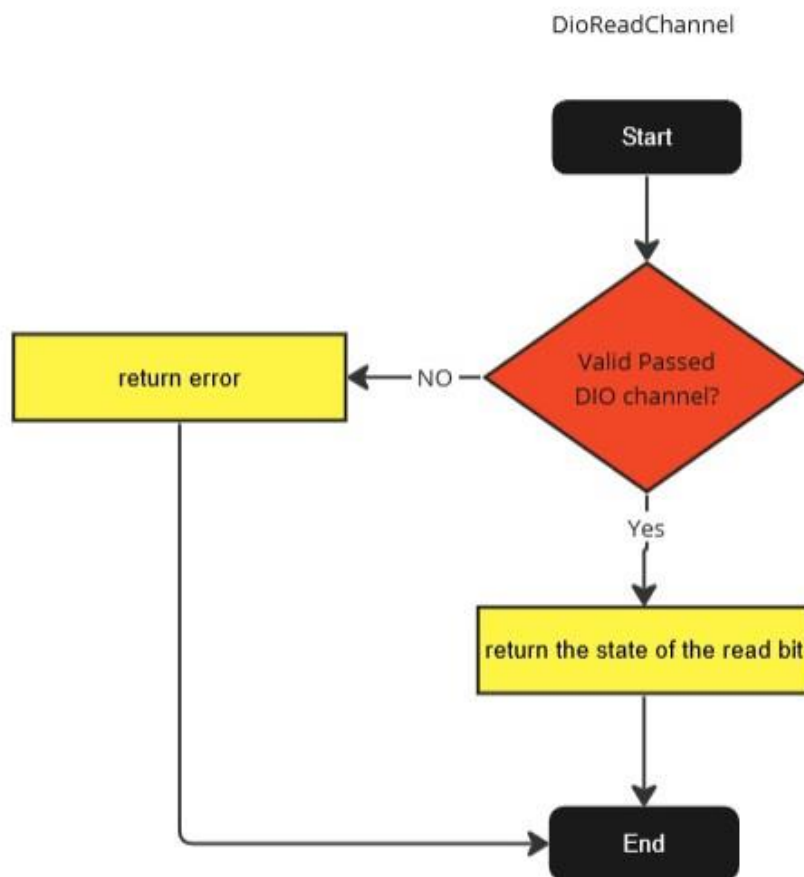
### Port driver:

Port init :

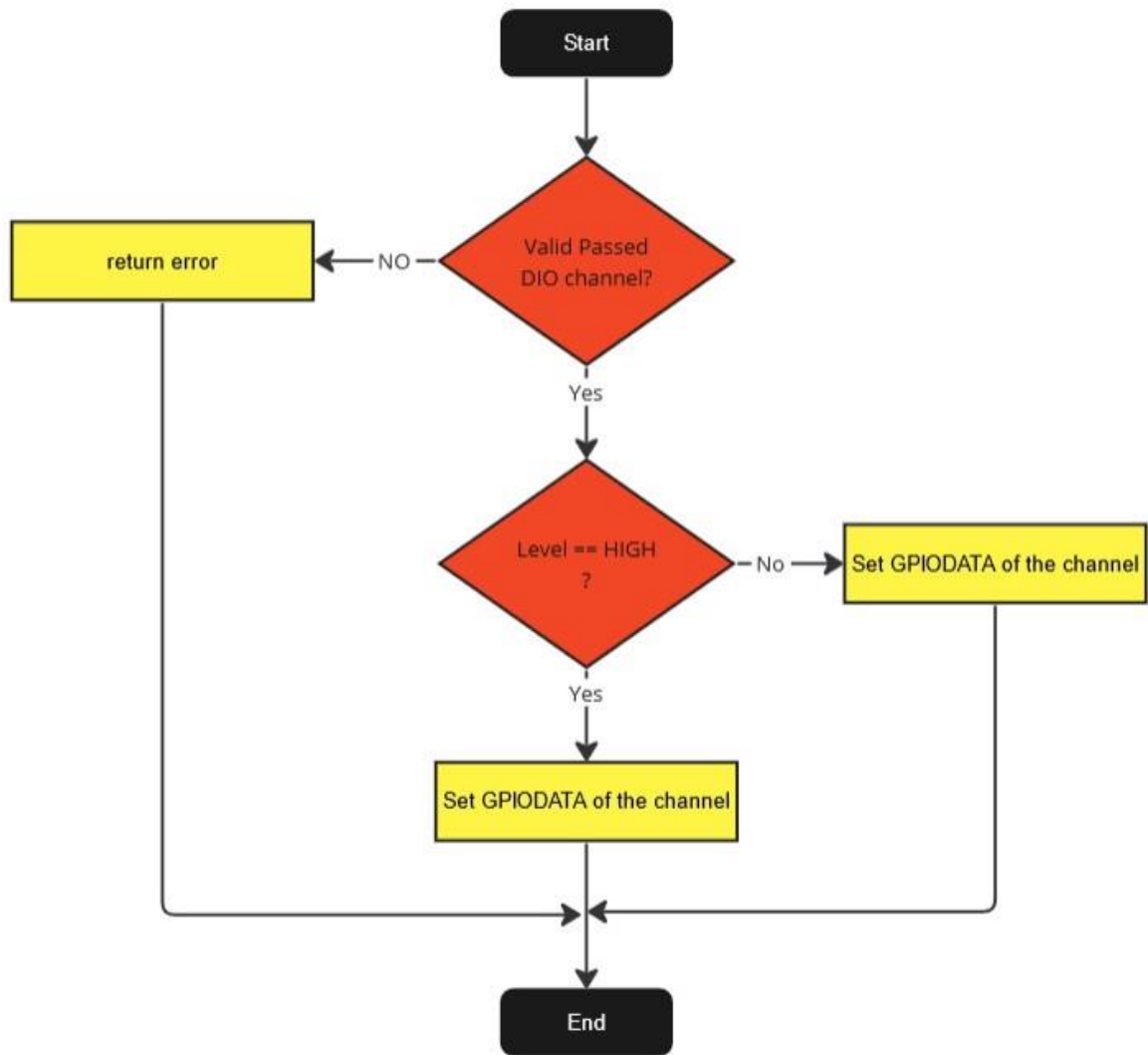




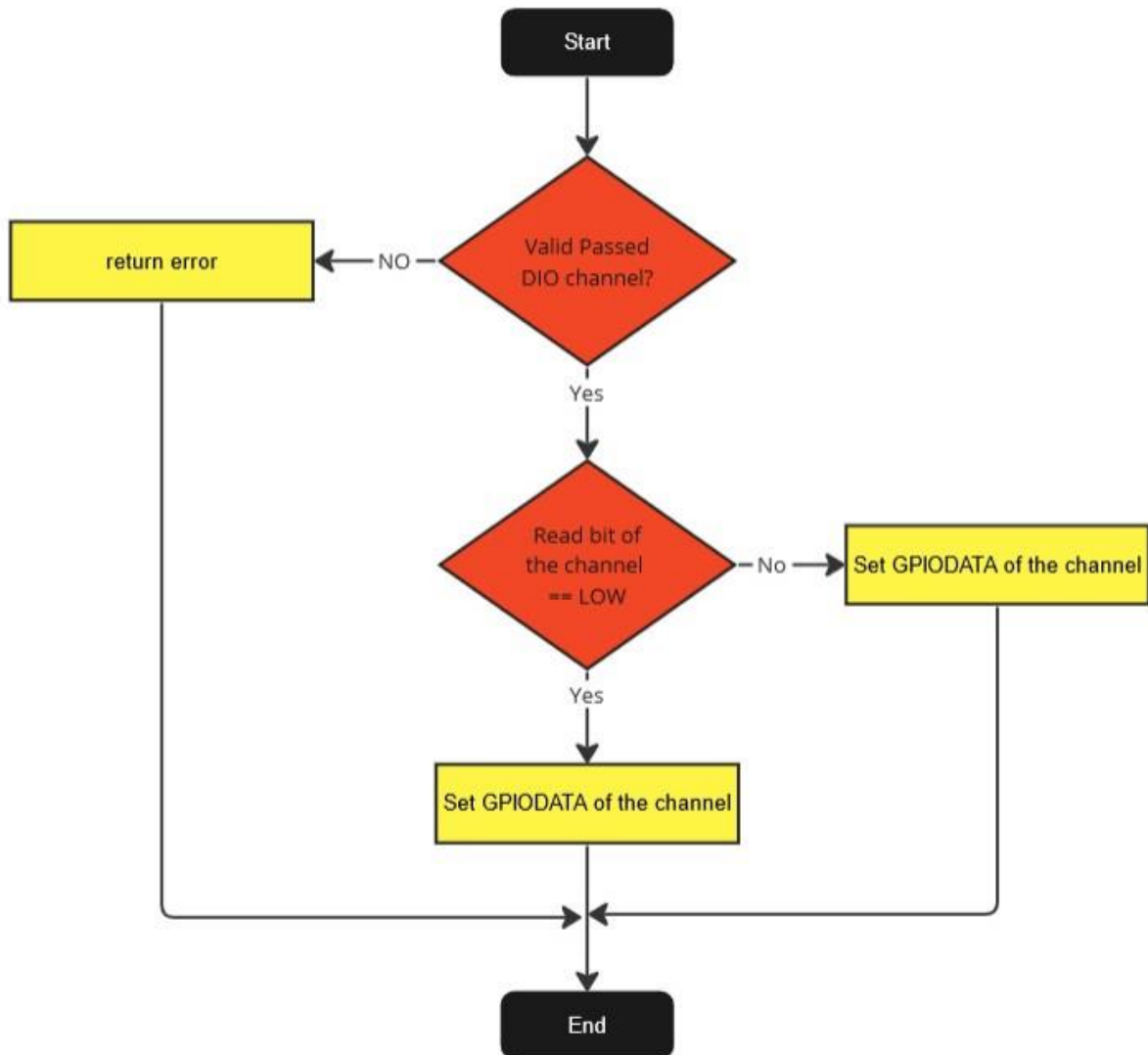
## DIO driver:



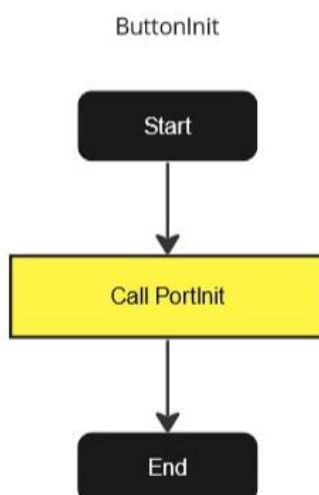
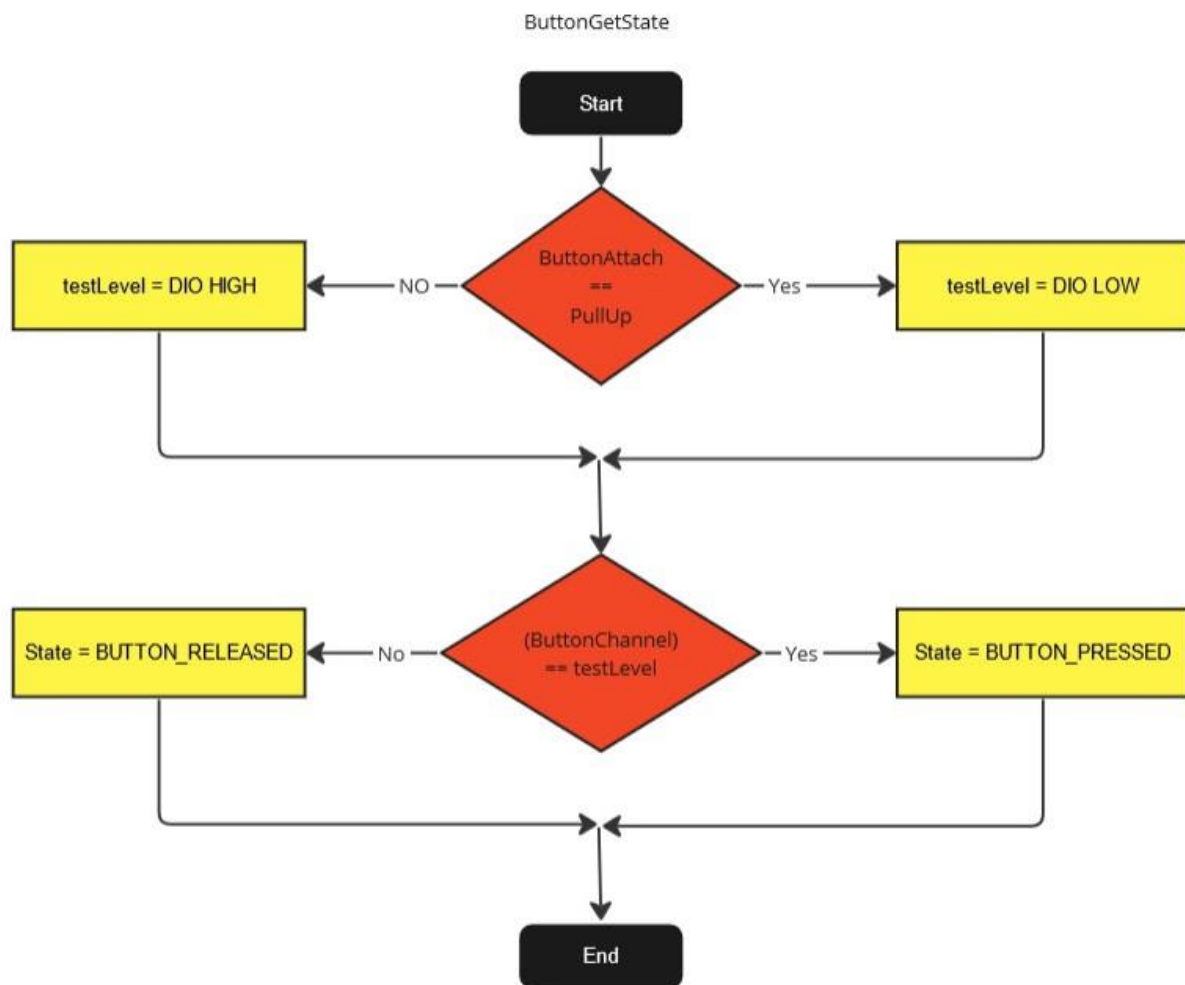
DioWriteChannel



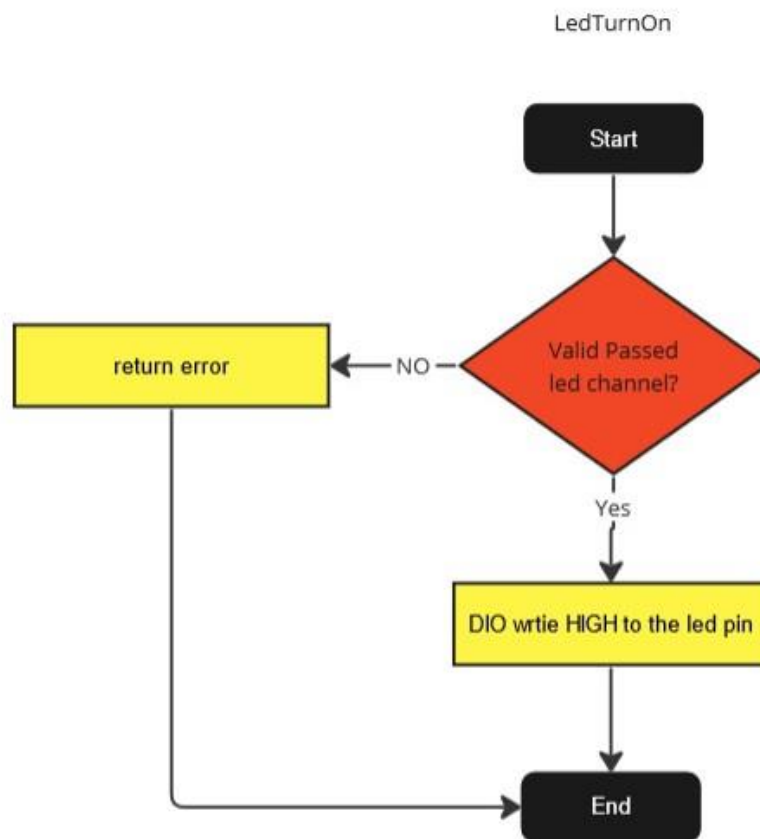
# DioToggleChannel



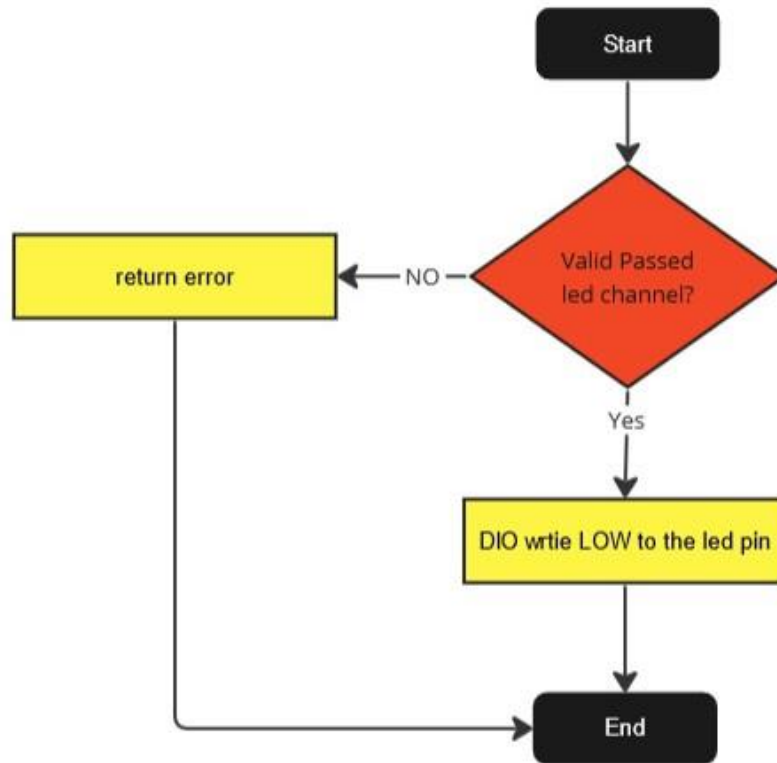
## Button driver



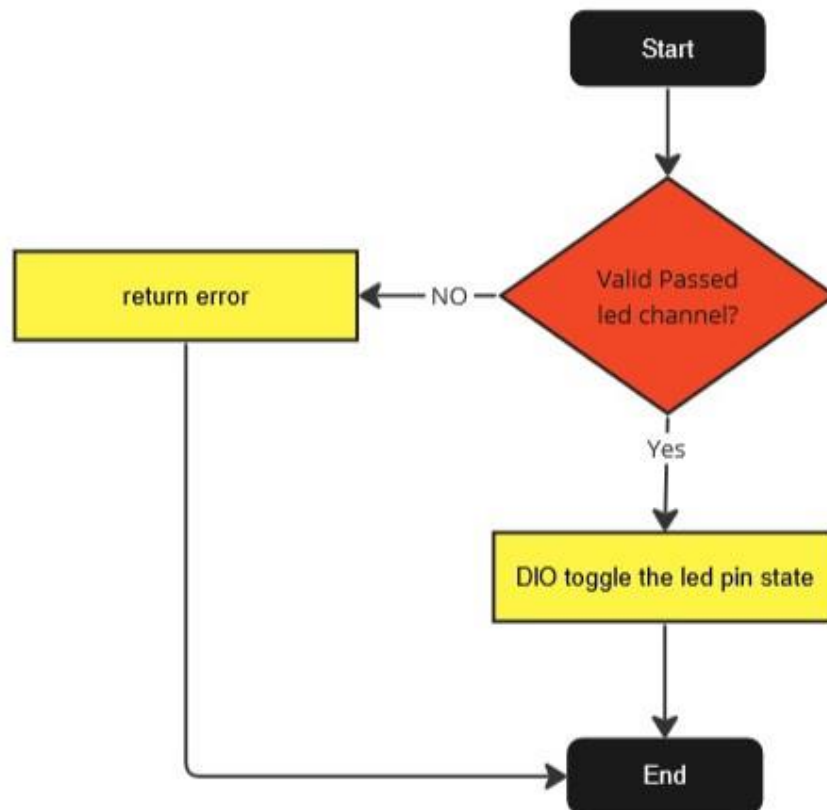
## LED driver:



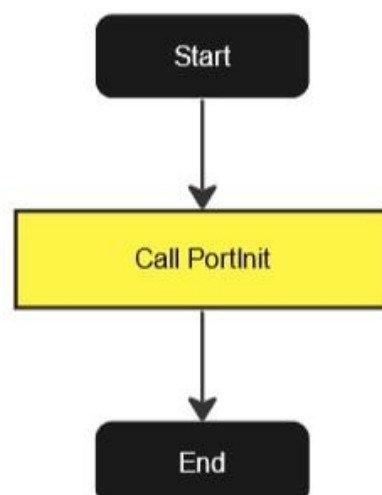
LedTurnOff



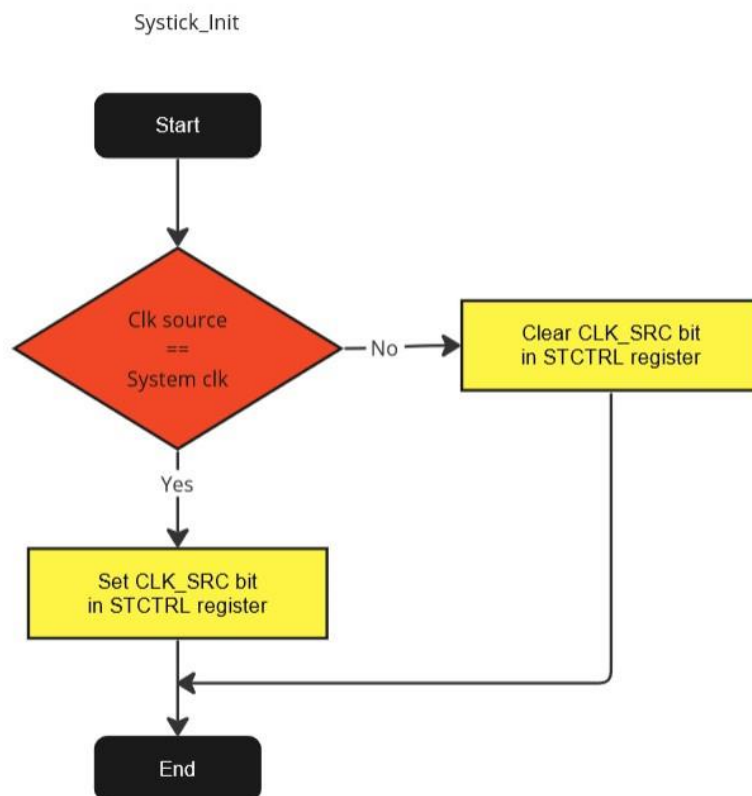
### LedToggle



### LedInit

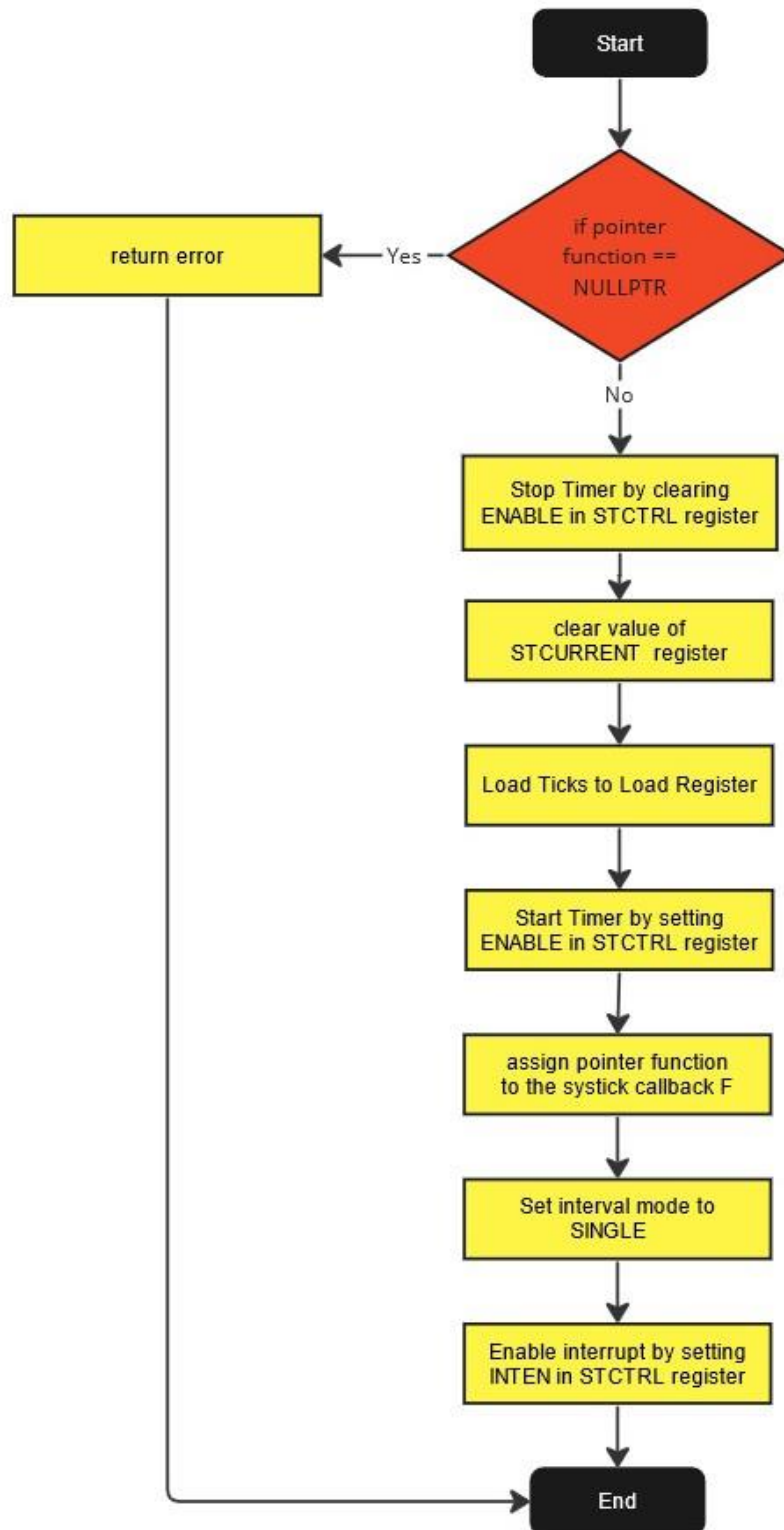


## Systick driver:

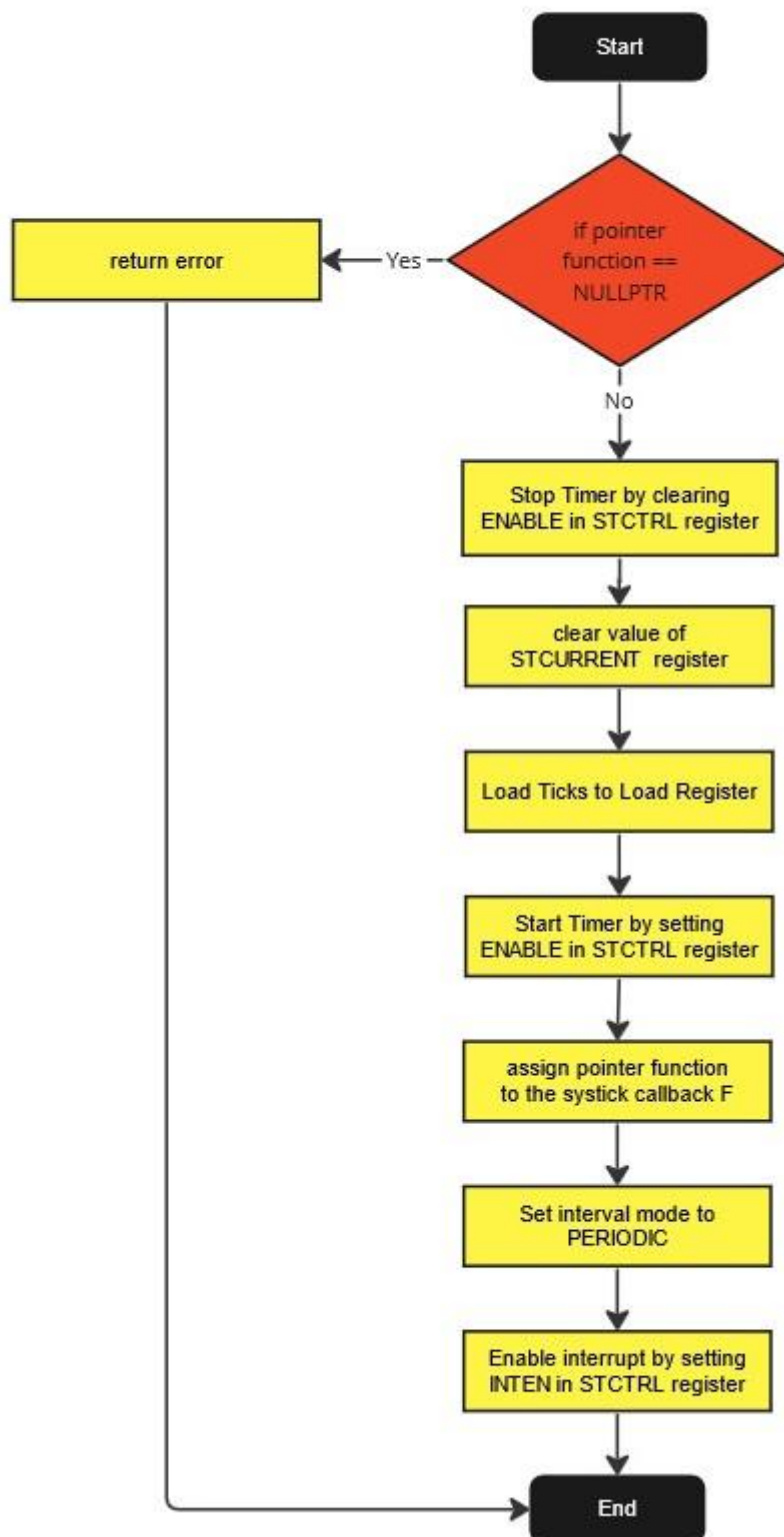




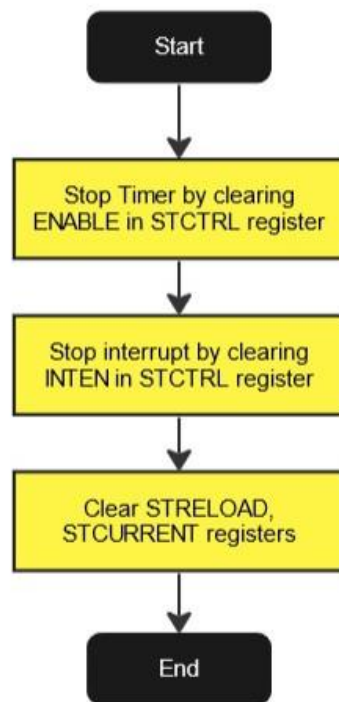
# Systick\_SetIntervalSingle



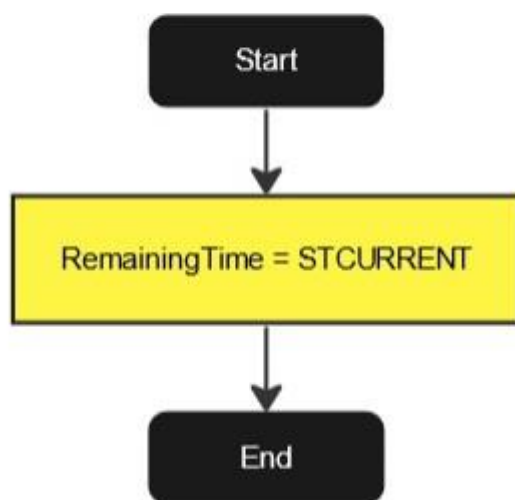
# Systick\_SetIntervalPeriodic



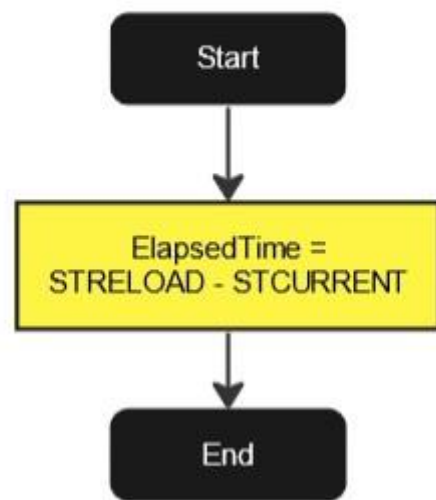
## Systick\_StopInterval



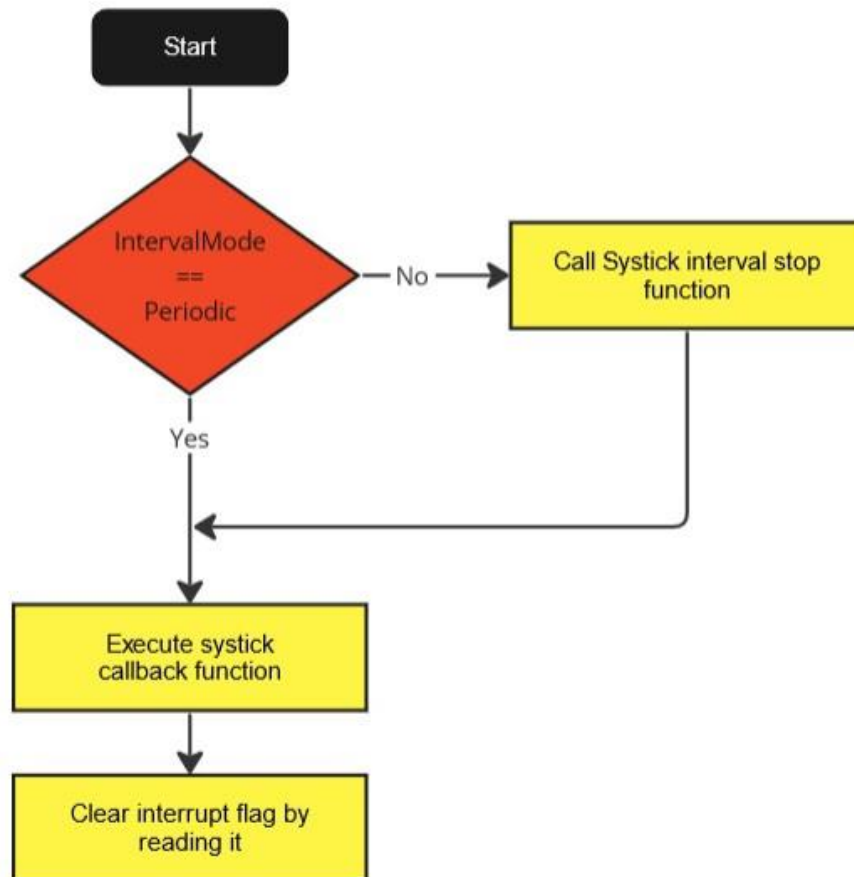
Systick\_GetRemainingTime



Systick\_GetElapsedTime



# Systick\_Handler



## 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 */
};
```



## Systick Cfg:-

```
#define      SYSTICK_CLK_SOURCE      SYSTEM_CLK  
static uint8  SYSTICK_u8IntervalMode =  SYSTICK_PERIODIC_INTERVAL;
```