DESIGN DOCUMENT

RGB LED CONTROL

V2.0 DESIGN

Team2:

-Amr El-Abd

-Ahmed Aatef

Table of Contents: -

Subject	Page

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

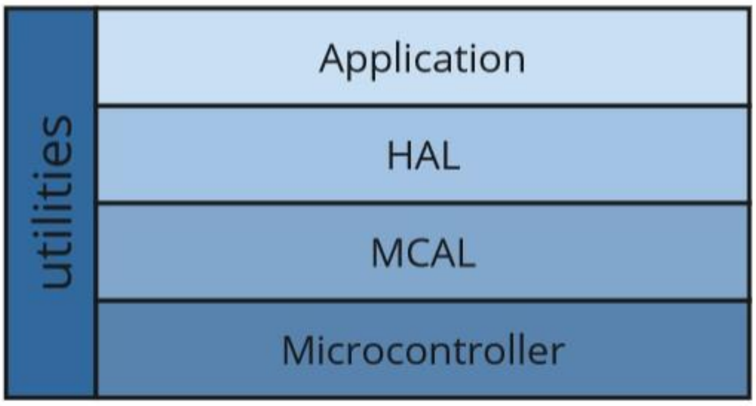
1. Use the TivaC board.
2. Use SW1 as an input button.
3. Use the RGB LED.

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.

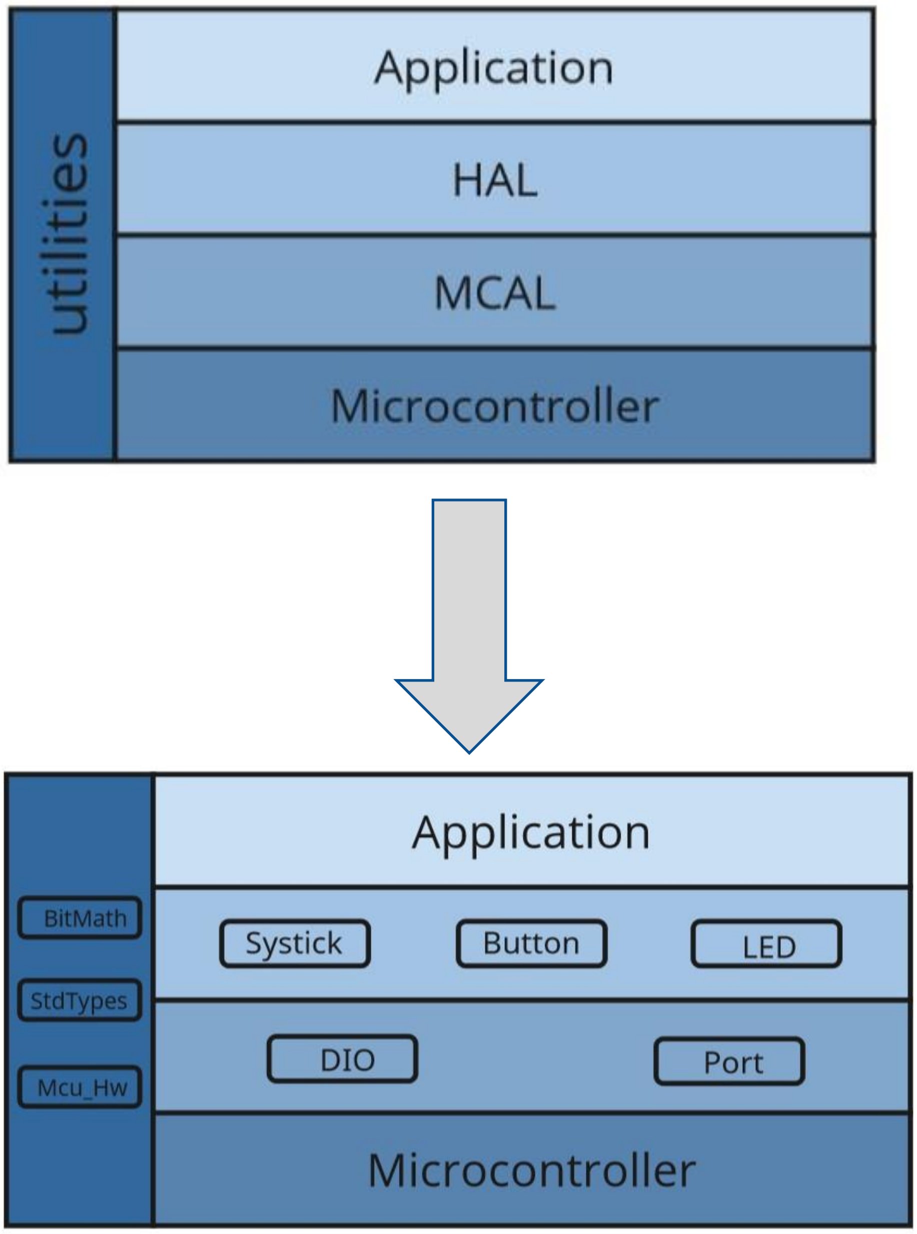
HAL Layer: This layer provides an abstraction for external devices connected to the microcontroller. The HAL layer provides interface to access external devices and hides the implementation details from the application layer.

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 pi

APIs:

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

DIO driver:

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

APIs:

```
enuDioLevel_t DioReadChannel(DioChannel_t ChannelId);

void 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:

```
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);

enuButtonState_t ButtonGetState(ButtonChannel_t ButtonChannel, enuButtonAttach_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:

enuErrorStatus_t Systick_Init (void);

enuErrorStatus_t Systick_SetIntervalSingle(uint32 u32Ticks, void(*Ptrf)(void));

enuErrorStatus_t Systick_SetIntervalPeriodic(uint32 u32Ticks, void(*Ptrf)(void));

enuErrorStatus_t Systick_StopInterval(void);

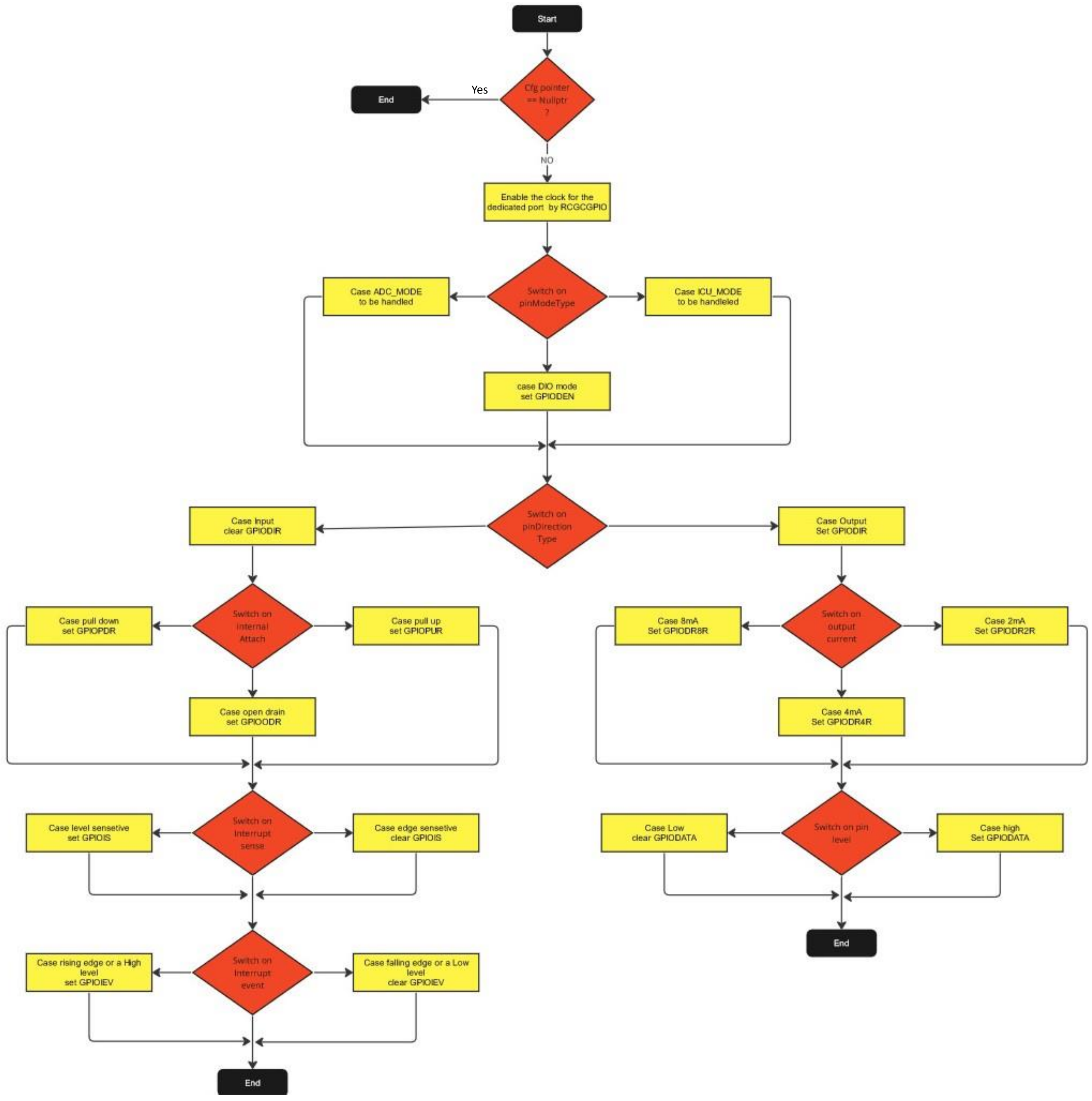
enuErrorStatus_t Systick_GetElapsedTime(uint32* u32ElapsedTime);

enuErrorStatus_t Systick_GetRemainingTime(uint32* u32RemainingTime);

Flowcharts for Functions:

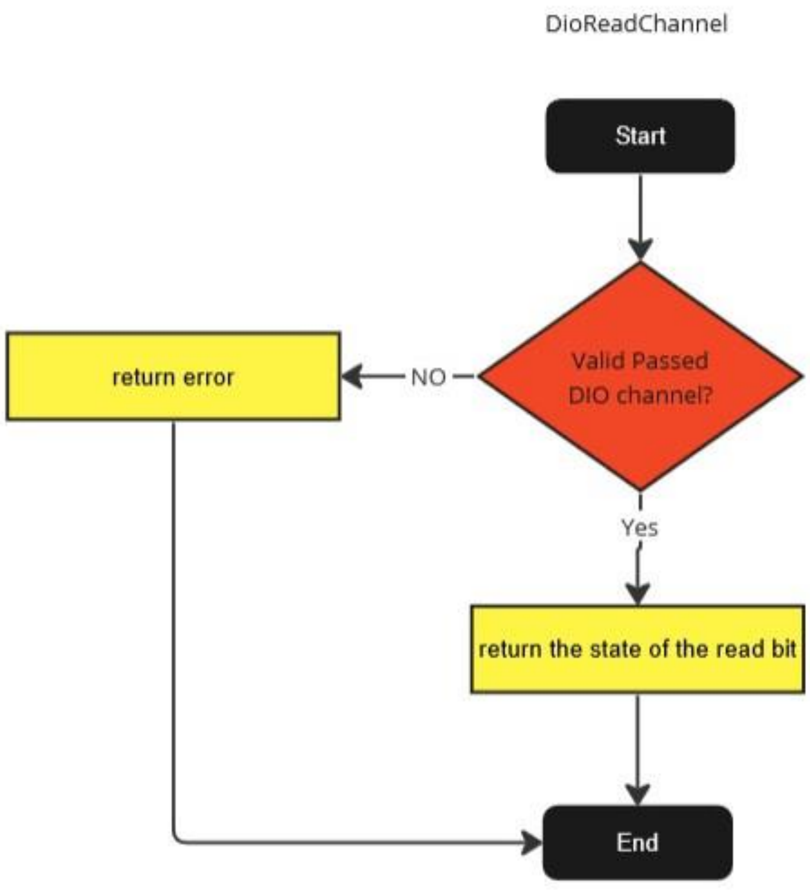
Port driver:

Port init :

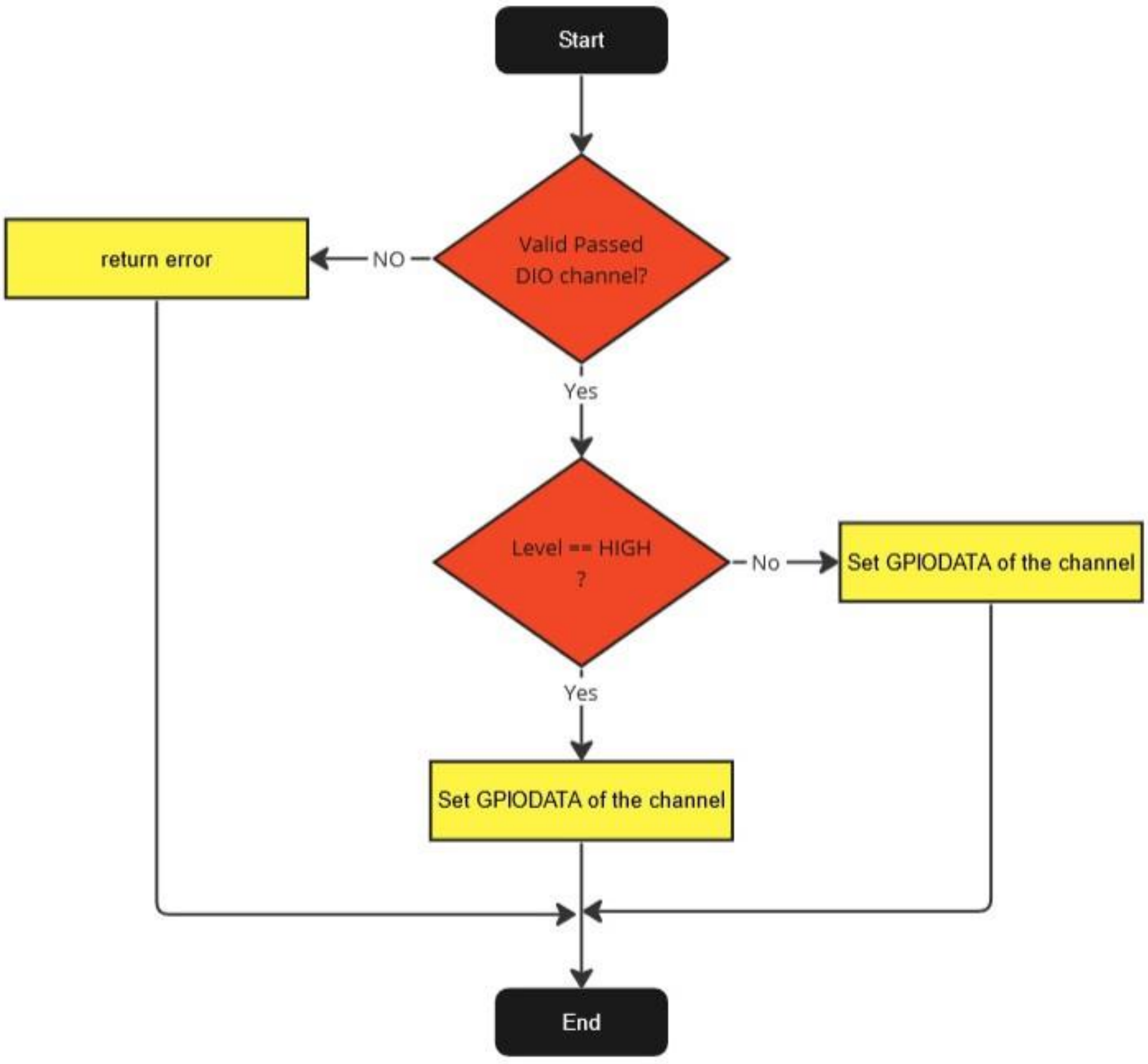


Start
Cfg pointer == Nullptr ?
Yes
End
NO
Enable the clock for the dedicated port by RCGCGPIO
Switch on pinModeType
Case ADC_MODE to be handled
Case ICU_MODE to be handleled
case DIO mode set GPIODEN
Switch on pinDirection Type
Case Input clear GPIODIR
Case Output Set GPIODIR
Switch on internal Attach
Case pull down set GPIOPDR
Case pull up set GPIOPUR
Case open drain set GPIOODR
Switch on Interrupt sense
Case level sensetive set GPIOIS
Case edge sensetive clear GPIOIS
Switch on Interrupt event
Case rising edge or a High level set GPIOIEV
Case falling edge or a Low level clear GPIOIEV
End
Switch on output current
Case 8mA Set GPIODR8R
Case 2mA Set GPIODR2R
Case 4mA Set GPIODR4R
Switch on pin level
Case Low clear GPIODATA
Case high Set GPIODATA
End

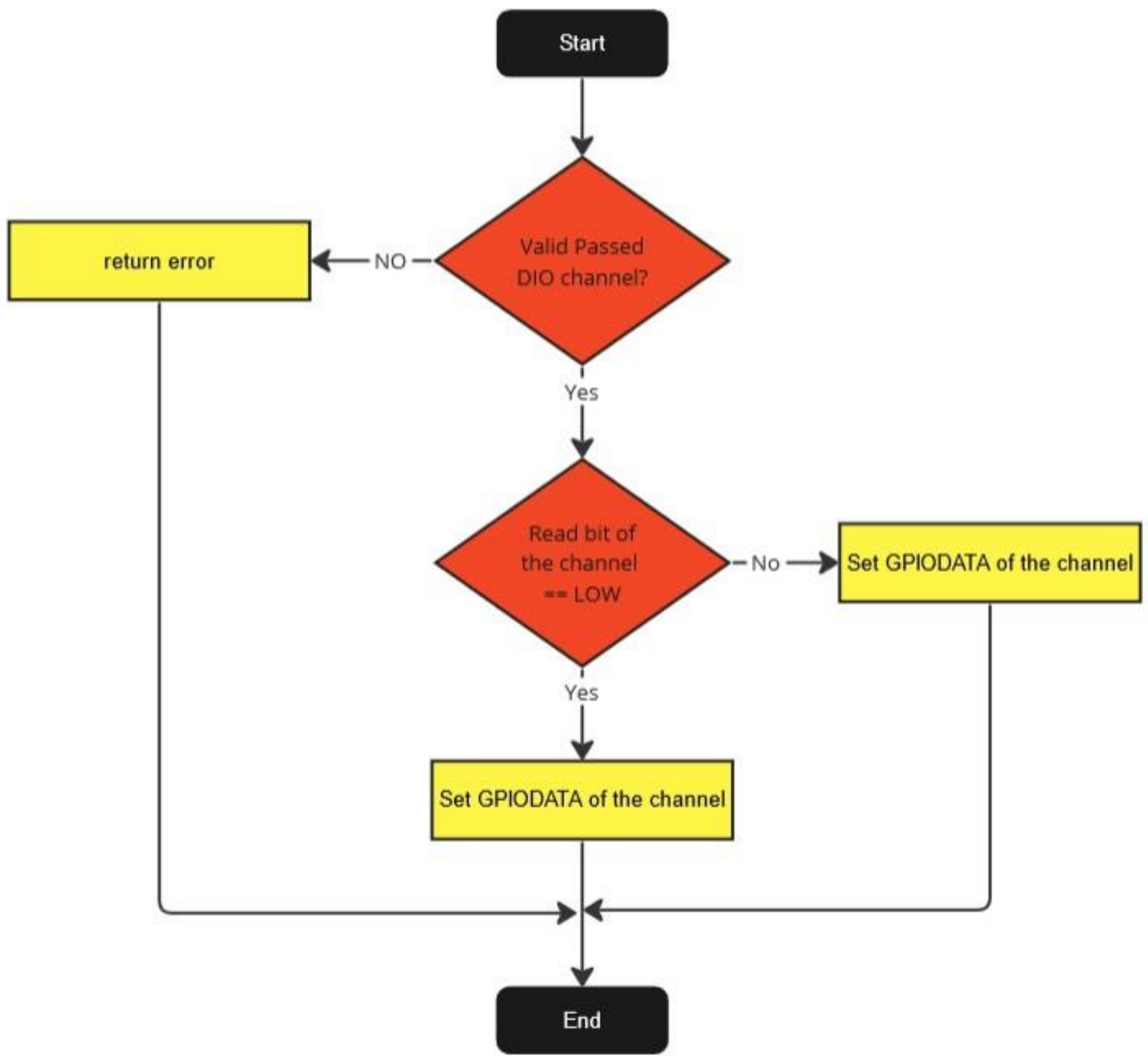
DIO driver:


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

DioWriteChannel

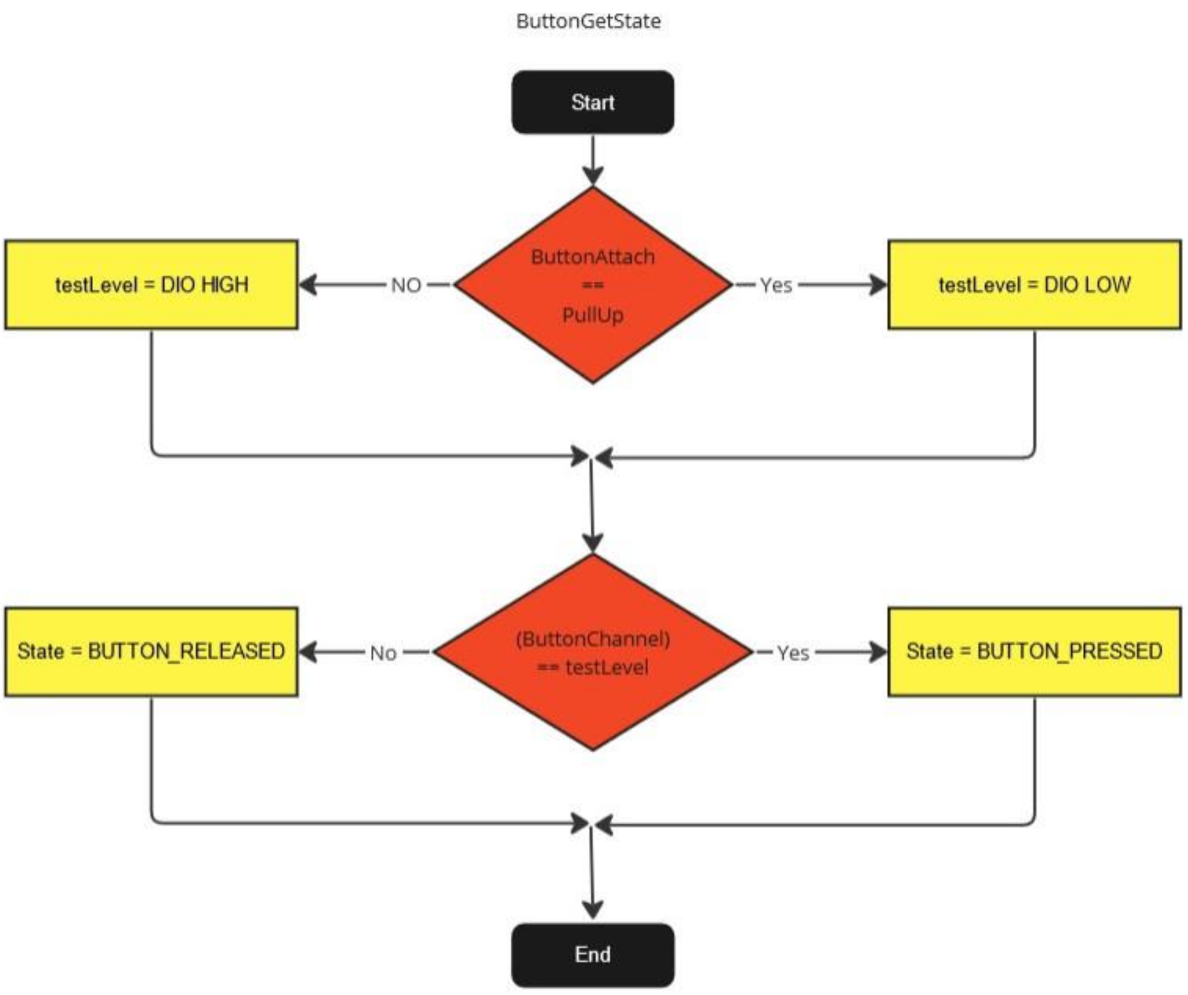

Start
Valid Passed DIO channel?
NO
return error
Yes
Level == HIGH ?
No
Set GPIODATA of the channel
Yes
Set GPIODATA of the channel
End

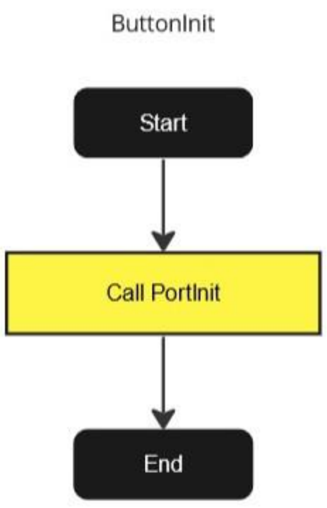
DioToggleChannel



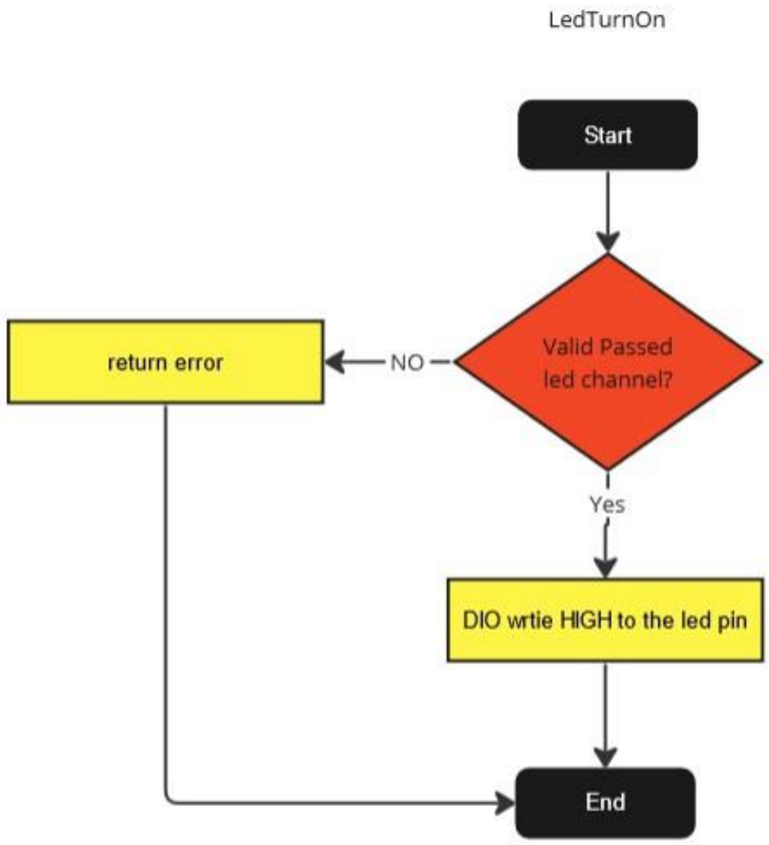
Start
Valid Passed DIO channel?
NO
return error
Yes
Read bit of the channel == LOW
No
Set GPIODATA of the channel
Yes
Set GPIODATA of the channel
End

Button driver

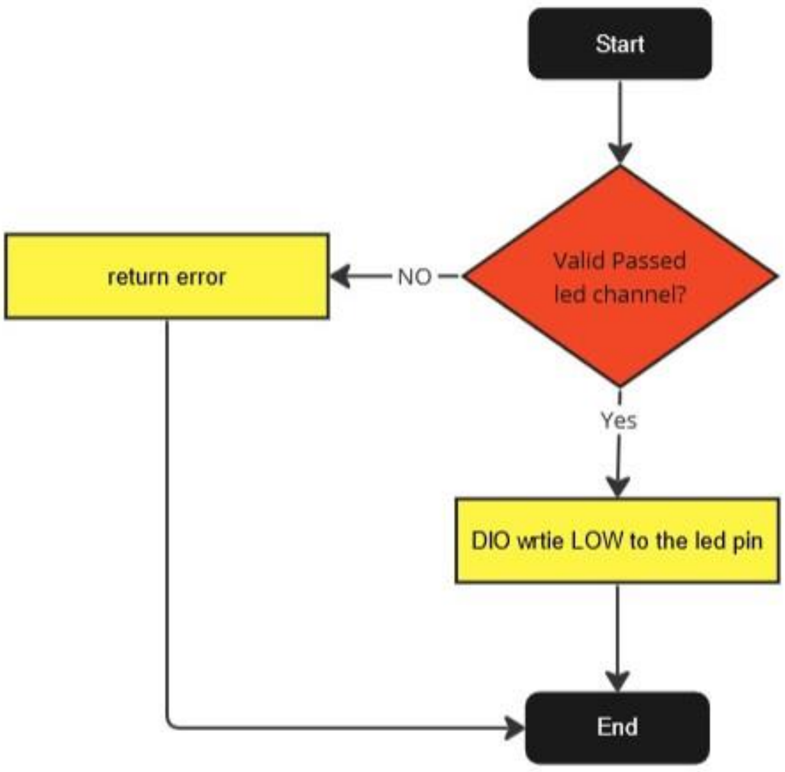




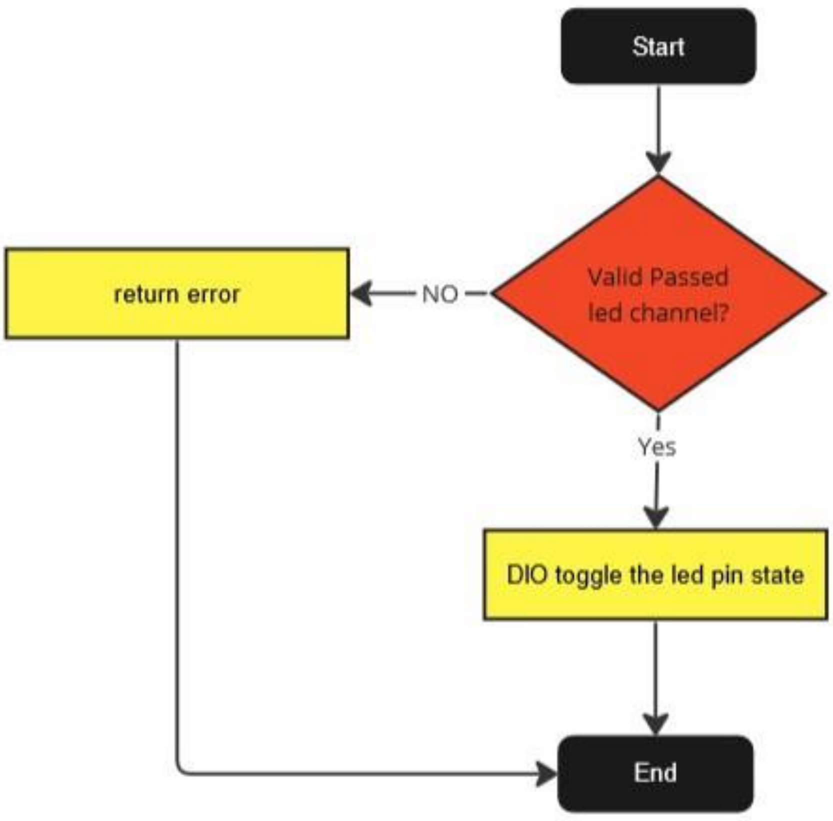
LED driver:


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

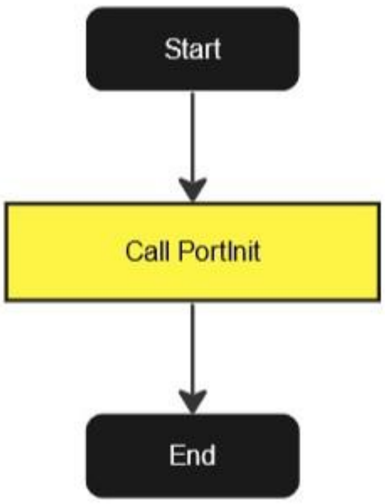
LedTurnOff


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

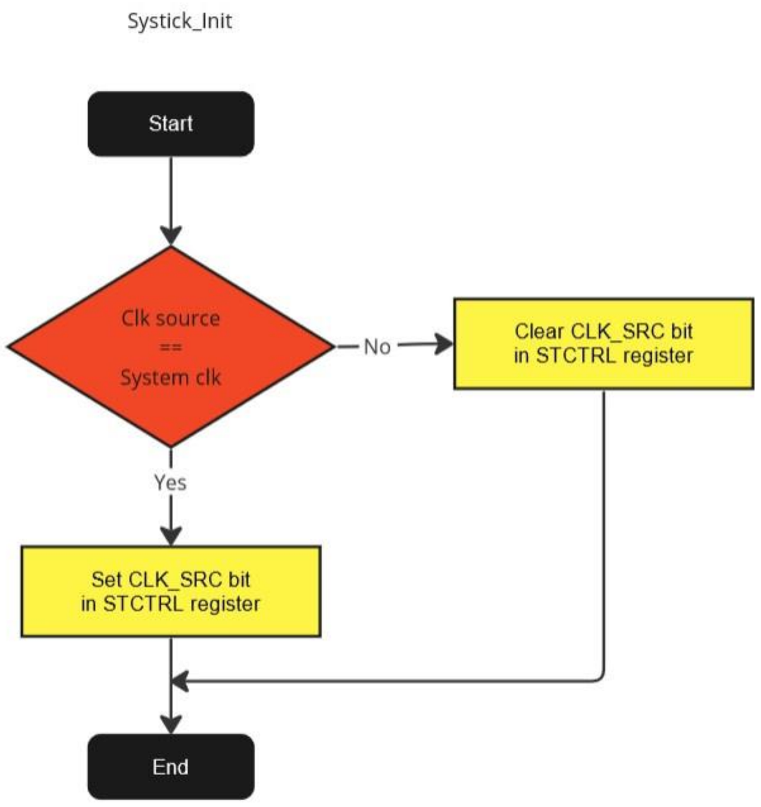
LedToggle



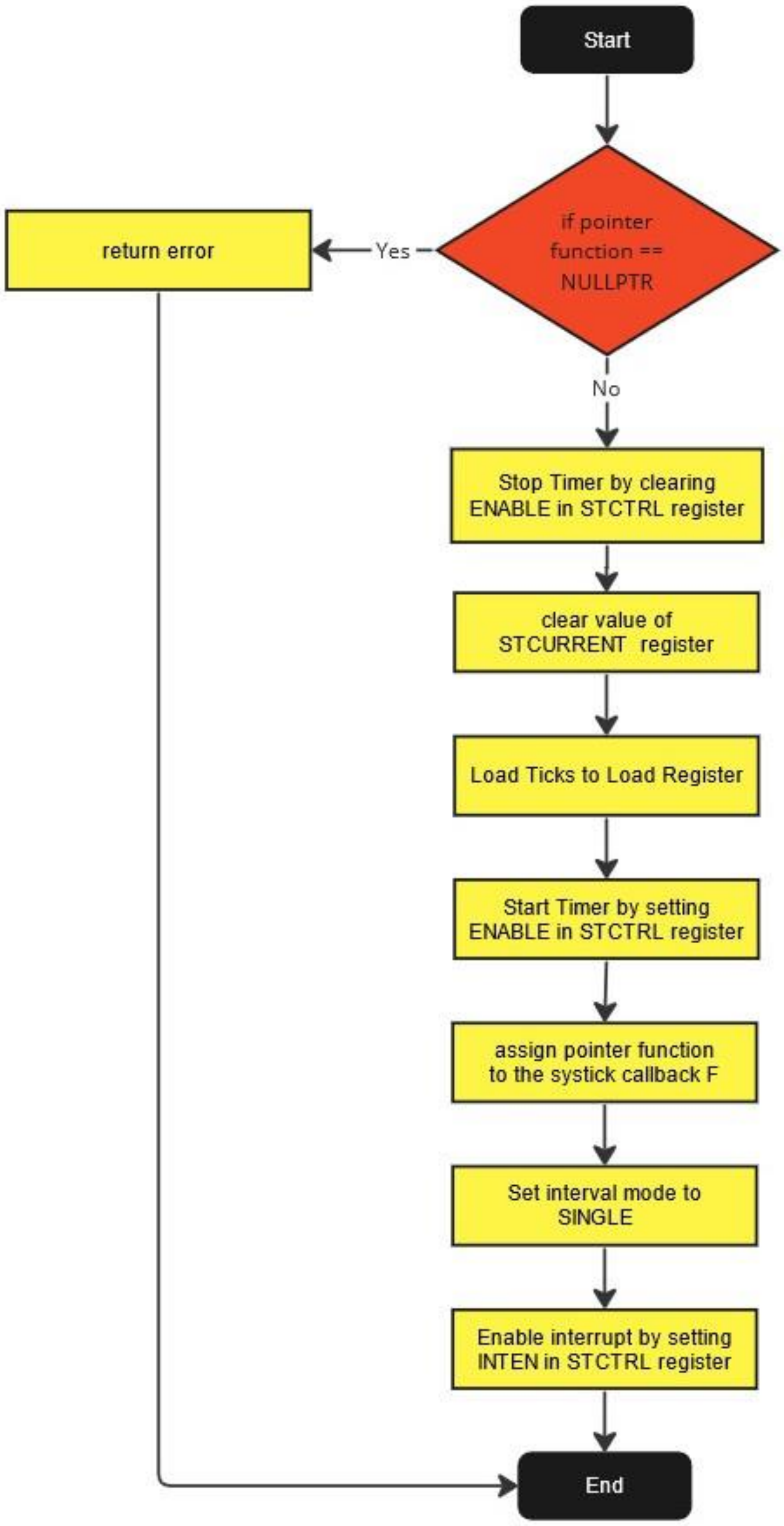
LedInit



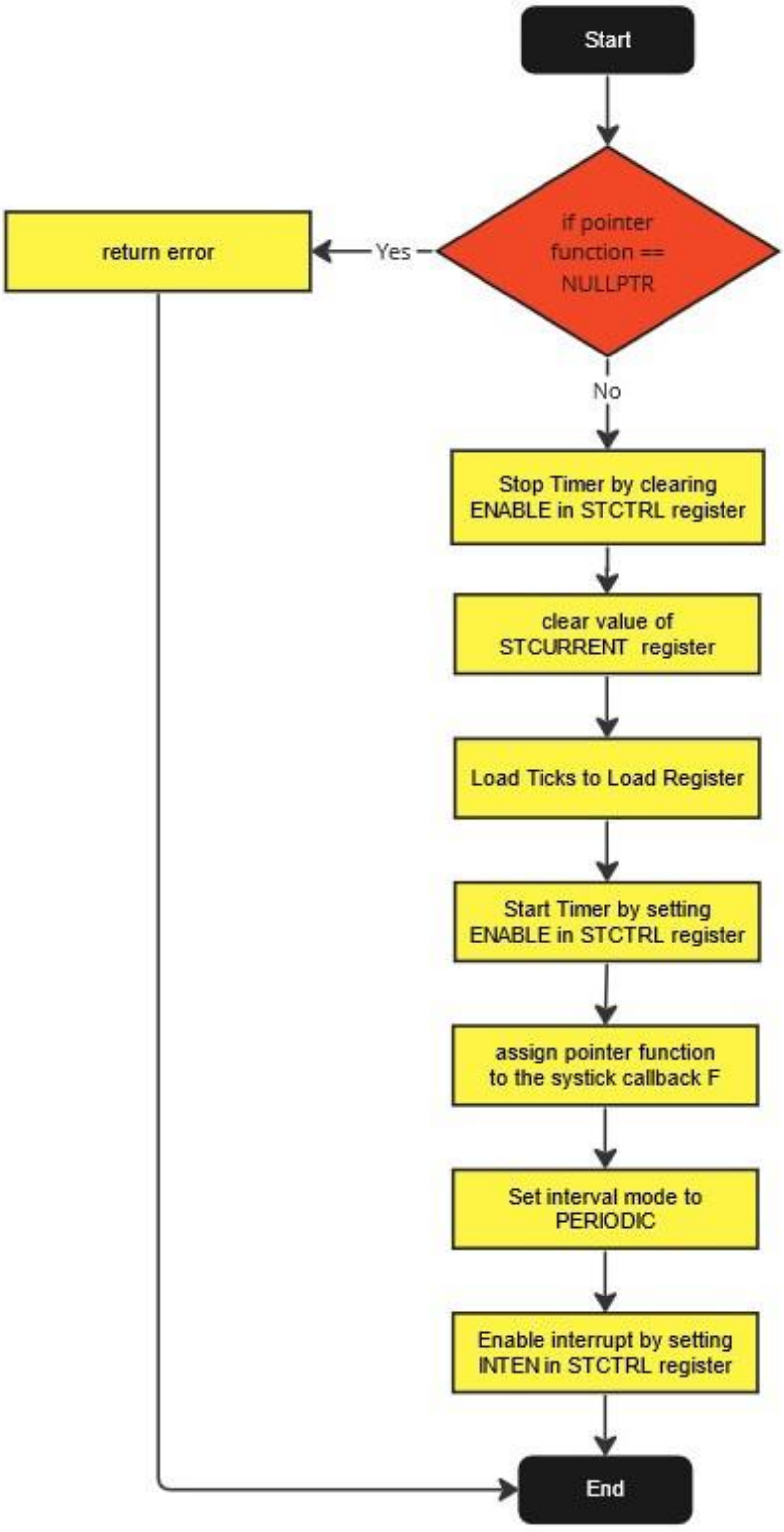
Systick driver:


Systick_Init
Start
Clk source
==
System clk
No
Clear CLK_SRC bit in STCTRL register
Yes
Set CLK_SRC bit in STCTRL register
End

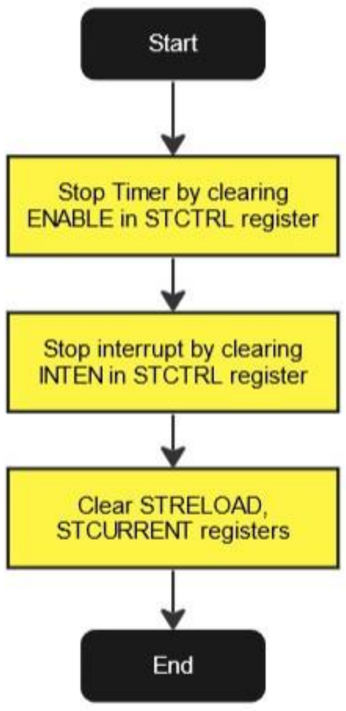
Systick_SetIntervalSingle


Start
if pointer function == NULLPTR
Yes
return error
No
Stop Timer by clearing ENABLE in STCTRL register
clear value of STCURRENT register
Load Ticks to Load Register
Start Timer by setting ENABLE in STCTRL register
assign pointer function to the systick callback F
Set interval mode to SINGLE
Enable interrupt by setting INTEN in STCTRL register
End

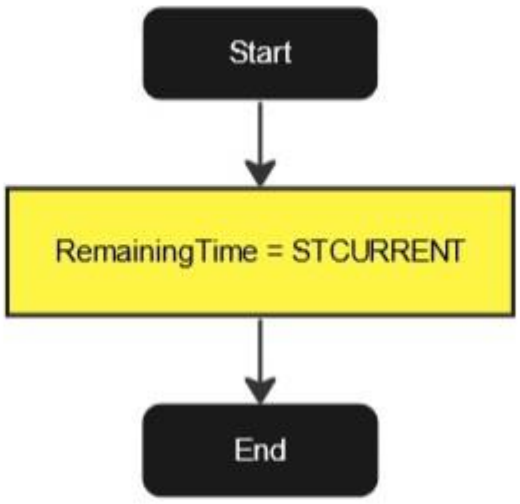
Systick_SetIntervalPeriodic


Start
if pointer function == NULLPTR
Yes
return error
No
Stop Timer by clearing ENABLE in STCTRL register
clear value of STCURRENT register
Load Ticks to Load Register
Start Timer by setting ENABLE in STCTRL register
assign pointer function to the systick callback F
Set interval mode to PERIODIC
Enable interrupt by setting INTEN in STCTRL register
End

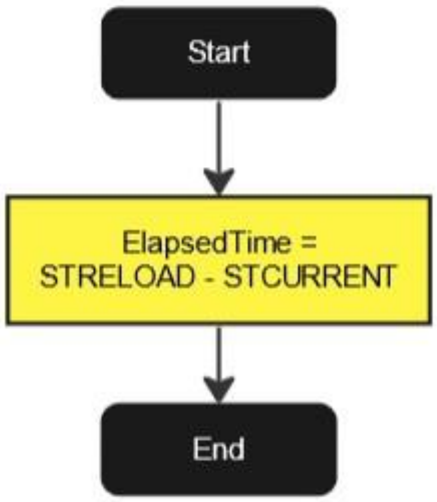
Systick_StopInterval

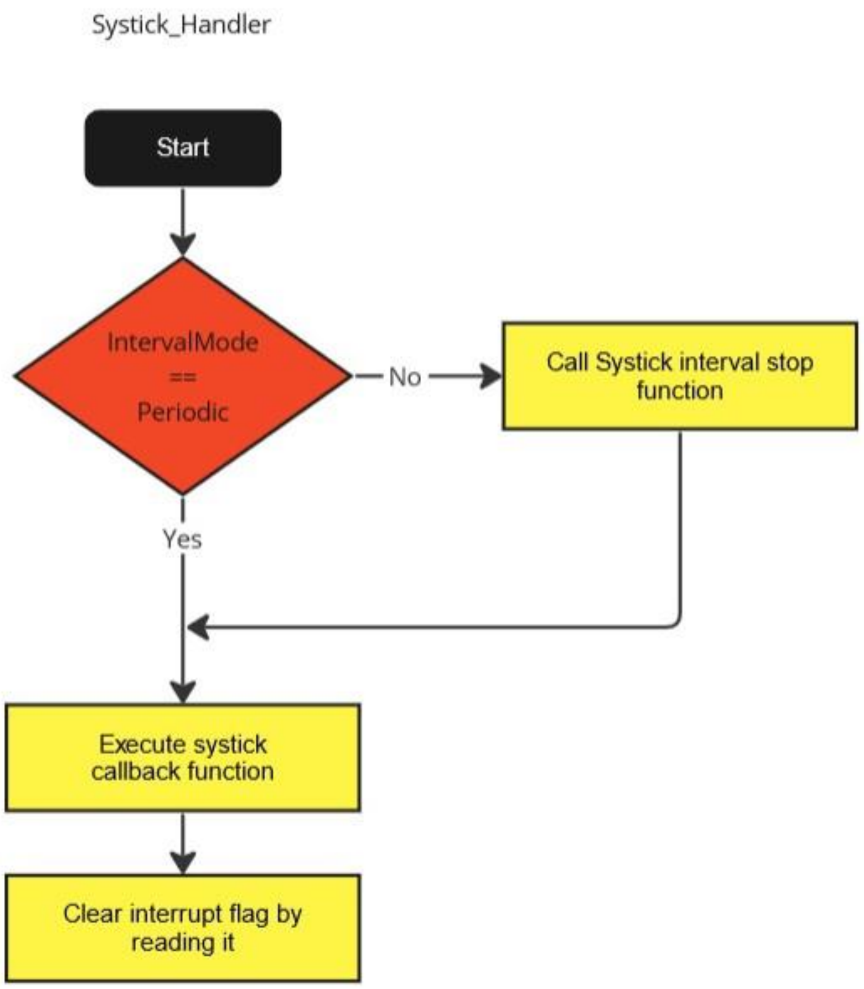

Start
Stop Timer by clearing ENABLE in STCTRL register
Stop interrupt by clearing INTEN in STCTRL register
Clear STRELOAD, STCURRENT registers
End

Systick_GetRemainingTime



Systick_GetElapsedTime




Systick_Handler
Start
IntervalMode
==
Periodic
No
Call Systick interval stop function
Yes
Execute systick callback function
Clear interrupt flag by reading it

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;
```