# LED Sequence V3.0

**By: Alaa Hisham**

---

## Project Description

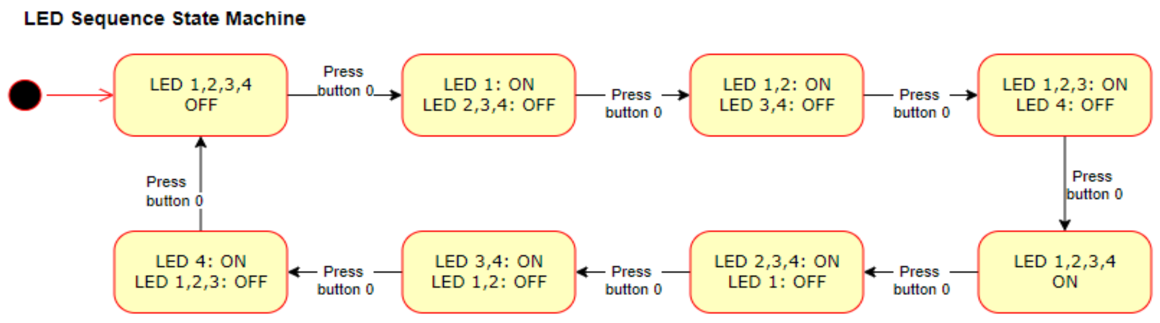
An App designed to display the following sequence:

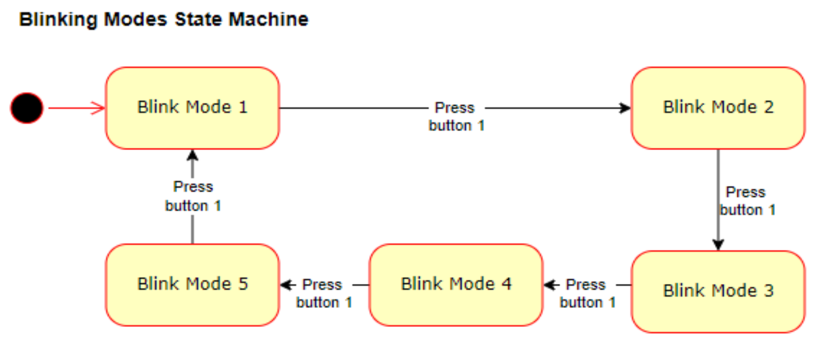
1. The sequence is described below
    1. Initially (OFF, OFF, OFF, OFF)
    2. Press 1 (BLINK_1, OFF, OFF, OFF)
    3. Press 2 (BLINK_1, BLINK_1, OFF, OFF)
    4. Press 3 (BLINK_1, BLINK_1, BLINK_1, OFF)
    5. Press 4 (BLINK_1, BLINK_1, BLINK_1, BLINK_1)
    6. Press 5 (OFF, BLINK_1, BLINK_1, BLINK_1)
    7. Press 6 (OFF, OFF, BLINK_1, BLINK_1)
    8. Press 7 (OFF, OFF, OFF, BLINK_1)
    9. Press 8 (OFF, OFF, OFF, OFF)
    10. Press 9 (BLINK_1, OFF, OFF, OFF)

2. When BUTTON1 has pressed the blinking on and off durations will be changed
    1. No press → **BLINK_1** mode (**ON**: 100ms, **OFF**: 900ms)
    2. First press → **BLINK_2** mode (**ON**: 200ms, **OFF**: 800ms)
    3. Second press → **BLINK_3** mode (**ON**: 300ms, **OFF**: 700ms)
    4. Third press → **BLINK_4** mode (**ON**: 500ms, **OFF**: 500ms)
    5. Fourth press → **BLINK_5** mode (**ON**: 800ms, **OFF**: 200ms)
    6. Fifth press → **BLINK_1** mode

The system is divided into layers and modules as follows.

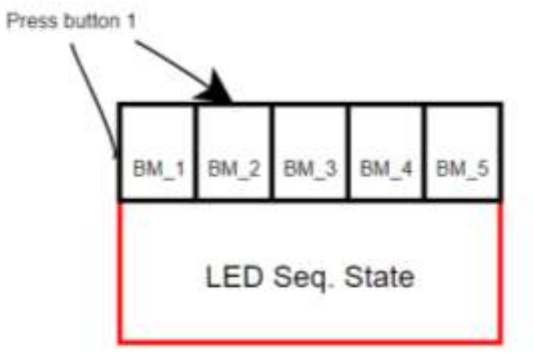
## Project State Machine





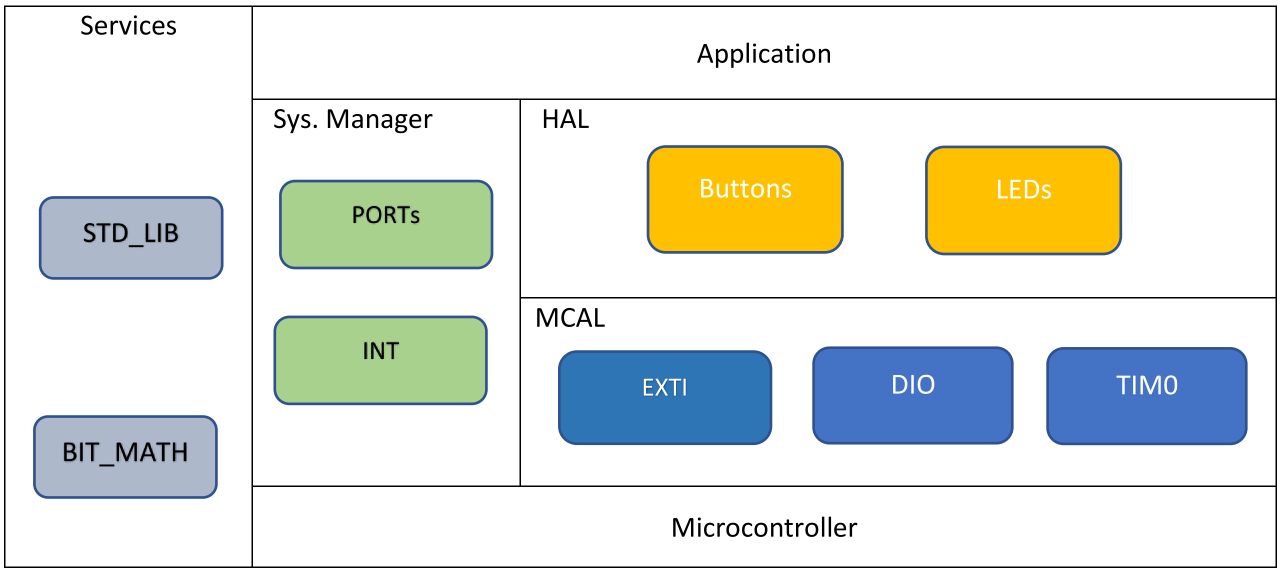
**Where, For Every State in LED Sequence:**

**(Where BM is Blink Mode)**



---

## Layered Architecture



## Modules & APIs

### System

- **Ports APIs:**

```
/**
@def configure the direction and initial value of all IO pins
*/
void Sys_PortInit(void);
```

- **INT APIs:**

```
/* Sets Global Interrupt Enable Bit   */
#define sei() __asm__ __volatile__ ("sei" ::: "memory")

/* Clears Global Interrupt Enable Bit */
#define cli() __asm__ __volatile__ ("cli" ::: "memory")
```

---

### MCAL

- **DIO APIs:**

```
/**
@def function to set the value of a single DIO pin
@param Copy_Port the port of the required pin
@param Copy_Pin the pin number in the given port
@param Copy_Value desired value (high or low) to set the pin to
@return error status
*/
EN_DIOErrorState_t DIO_SetPinVal(uint8_t Copy_Port,uint8_t Copy_Pin, uint8_t
Copy_Value);

/**
@def function to configure the value of an entire port
@param Copy_Port the desired port
@param Copy_Value desired 8-bit value to set the port to
@return error status
*/
EN_DIOErrorState_t DIO_SetPortVal(uint8_t Copy_Port, uint8_t Copy_Value);

/**
@def Set or clear multiple pins on port without affecting the rest of the pins
@param Copy_Port the port of the required pin
@param Copy_portMask 8-bit value where the desired pins are represented by ones
@param Copy_Value desired value (high or low) to set the pins to
@return error status
*/
EN_DIOErrorState_t DIO_MaskPortVal(uint8_t Copy_Port, uint8_t Copy_portMask, uint8_t
Copy_Value);
```

- **EXTI APIs:**

```
/**
Initializes given External interrupt with given mode
*/
EN_EXTIErrorState_t EXTI_Init(EN_EXTI_t Copy_Int, EN_EXTISenseMode_t Copy_Mode);

/**
enables specific interrupt of given EXTI number
*/
EN_EXTIErrorState_t EXTI_Enable(EN_EXTI_t Copy_Int);

/**
disables specific interrupt of given EXTI number
*/
EN_EXTIErrorState_t EXTI_Disable(EN_EXTI_t Copy_Int);

/**
Sets given function to be called when given EXTI is triggered
*/
EN_EXTIErrorState_t EXTI_SetCallback(EN_EXTI_t Copy_Int, void
(*Copy_pCallbackFn)(void));
```

- **TIMER APIs:**

```
/*****************************************************************/
/* Initialize the timer mode                                     */
/*****************************************************************/
void TIM0_voidInit();

/*****************************************************************/
/* Start the timer clock after prescaling it with given value    */
/*****************************************************************/
EN_TIMErrorState_t TIM0_Start(uint8_t Copy_prescaler);

/*****************************************************************/
/* Stop the timer
*/
/*****************************************************************/
void TIM0_Stop();

/*****************************************************************/
/* Set a value for the timer to start from                       */
/*****************************************************************/
void TIM0_SetValue(uint8_t Copy_Value);

/*****************************************************************/
/* Generate delay (busy waiting)                                 */
/*****************************************************************/
EN_TIMErrorState_t TIM0_SyncDelay(uint32_t Copy_delayTime, en_timeUnits_t
Copy_timeUnit);
```

## HAL

- ### LEDs APIs

```
/**
 * enables displaying output on given led
*/
EN_LEDErrorState_t LED_EnableLED(ST_LED* Copy_LED);

/**
 * Disables displaying output on given led
 */
EN_LEDErrorState_t LED_DisableLED(ST_LED* Copy_LED);

/**
 * Set the state of the given led to On/Off
 */
EN_LEDErrorState_t LED_setState(ST_LED* Copy_LED, EN_LEDState Copy_LEDState);

/**
 * Set the state of multiple LEDs on the same port
 * without affecting the rest of its pins
 */
void LED_setLEDPortState(uint8_t Copy_port, uint8_t Copy_portValue, EN_LEDState
Copy_LEDsState);

/**
 * Blink led with given on and off time
 */
EN_LEDErrorState_t LED_Blink(ST_LED* Copy_Led, u_int_16 Copy_OnTime, u_int_16
Copy_OffTime);

/**
 * Blink multiple LEDs on a port with given on and off time
 */
EN_LEDErrorState_t LED_BlinkPort(uint8_t Copy_port, uint8_t Copy_portPins,
u_int_16 Copy_OnTime, u_int_16 Copy_OffTime);
```

- ### Buttons APIs

```
/**
enables reading from given switch
*/
EN_SWError_t SW_EnableSwitch(ST_Switch* Copy_Switch);

/**
Disables reading from given switch
*/
EN_SWError_t SW_DisableSwitch(ST_Switch* Copy_Switch);


/**
Initialize switch as External Interrupt source
*/
EN_SWError_t SW_EXTIMode(ST_Switch* Copy_Switch, EN_SW_Interrupt_t Copy_IntEvent,
void (*Copy_pvCallbackFn)(void));
```

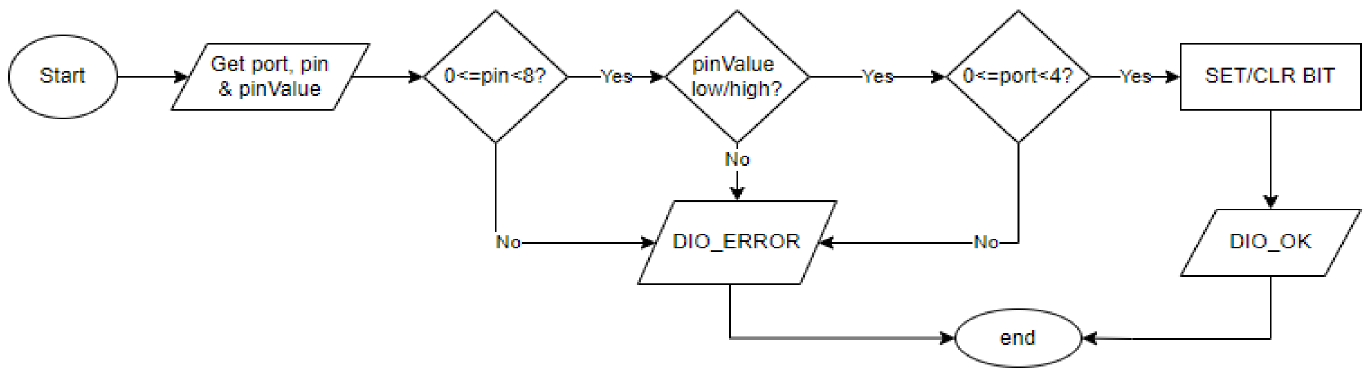
## APPLICATION

```
/**
initialize the configured io ports
Enable General Interrupts
*/
void App_Init(void);

/**
The app main logic
*/
void App(void);


/************** Buttons Callback Functions *************/
/**
increments the count of button presses to enter the
right state
*/
void Button0_Callback(void);

/**
Increments Blink Mode with each press
and resets it if it exceeds max value
*/
void Button1_Callback(void);
```

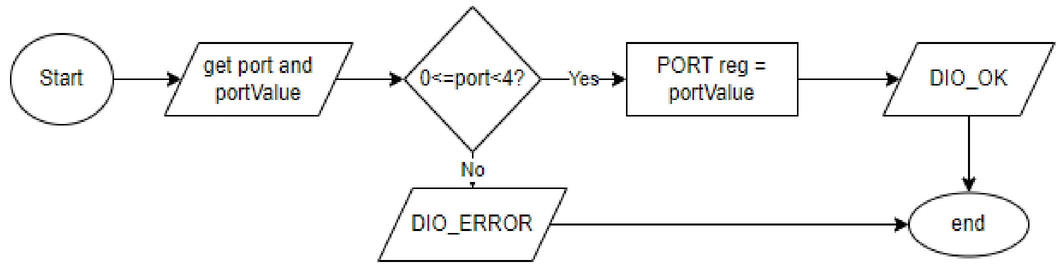
## API Design Diagrams

- **DIO APIs**

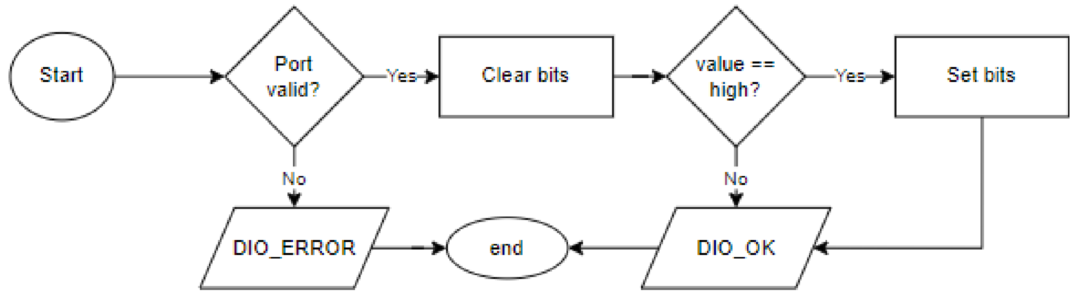
### DIO_SetPinVal



### DIO_SetPortVal

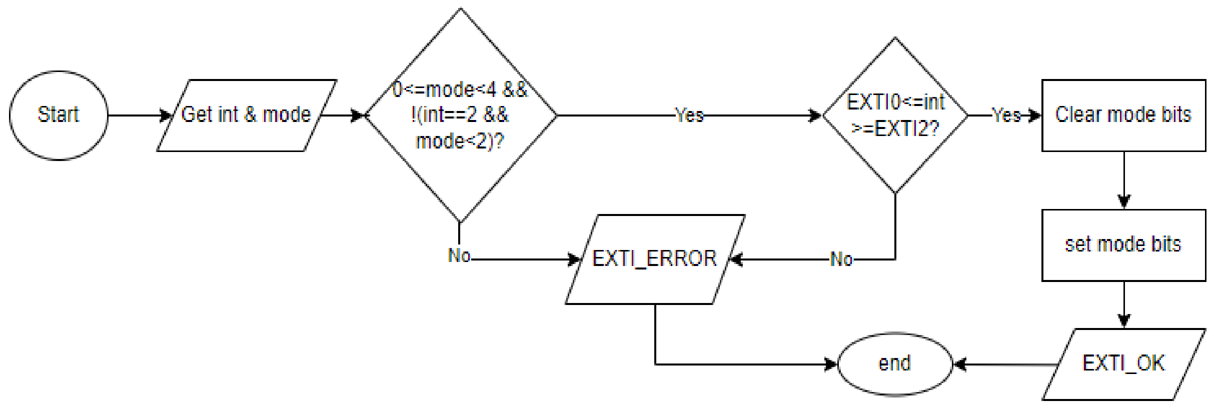


### DIO_ MaskPortVal

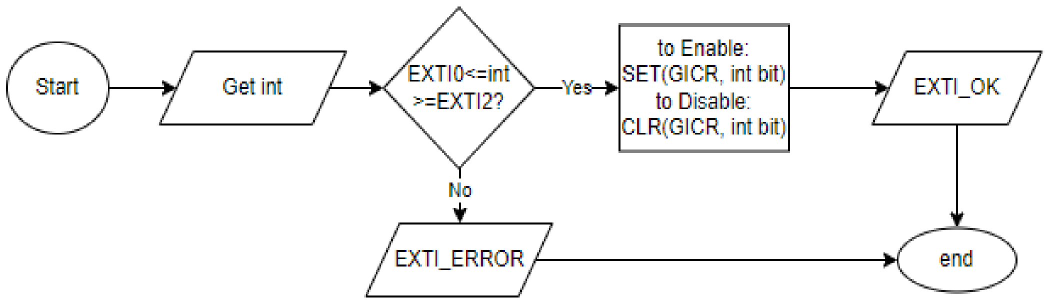


- **EXTI APIs**

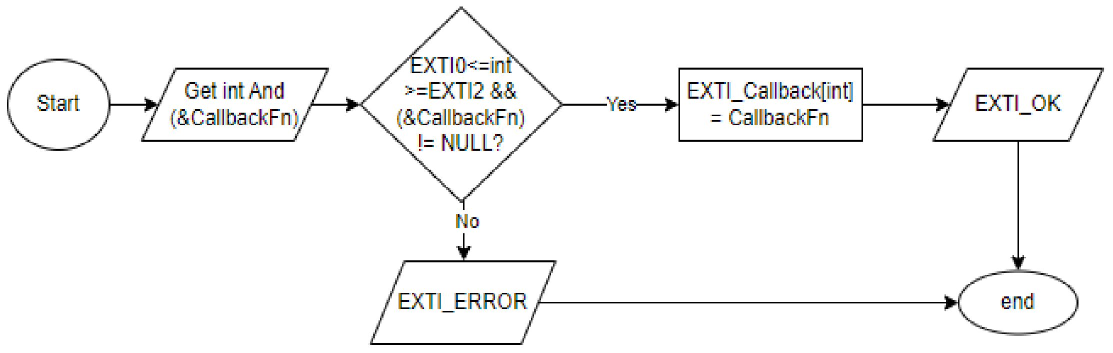
## EXTI_Init



## EXTI_Enable & EXTI_Disable

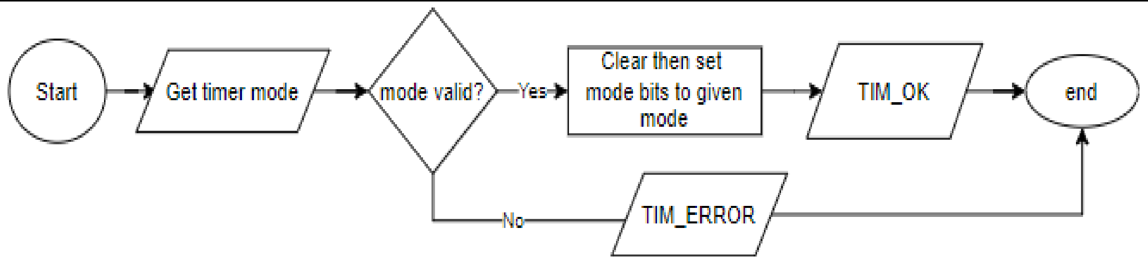


## EXTI_SetCallback

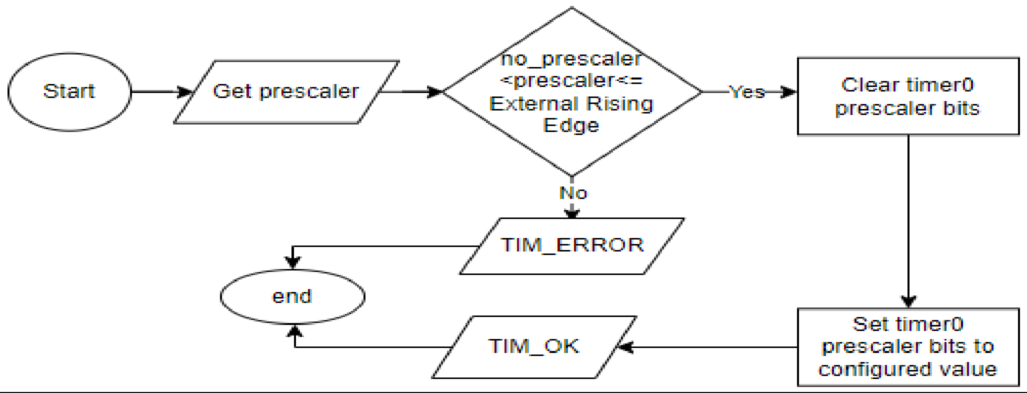

Start
Get int And (&CallbackFn)
EXTI0<=int >=EXTI2 && (&CallbackFn) != NULL?
Yes
No
EXTI_Callback[int] = CallbackFn
EXTI_OK
EXTI_ERROR
end

- **TIMER APIs**

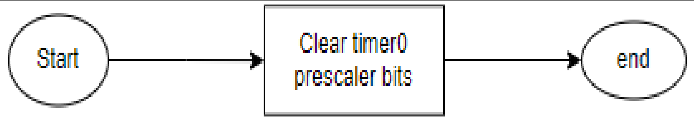
## TIM0_Init



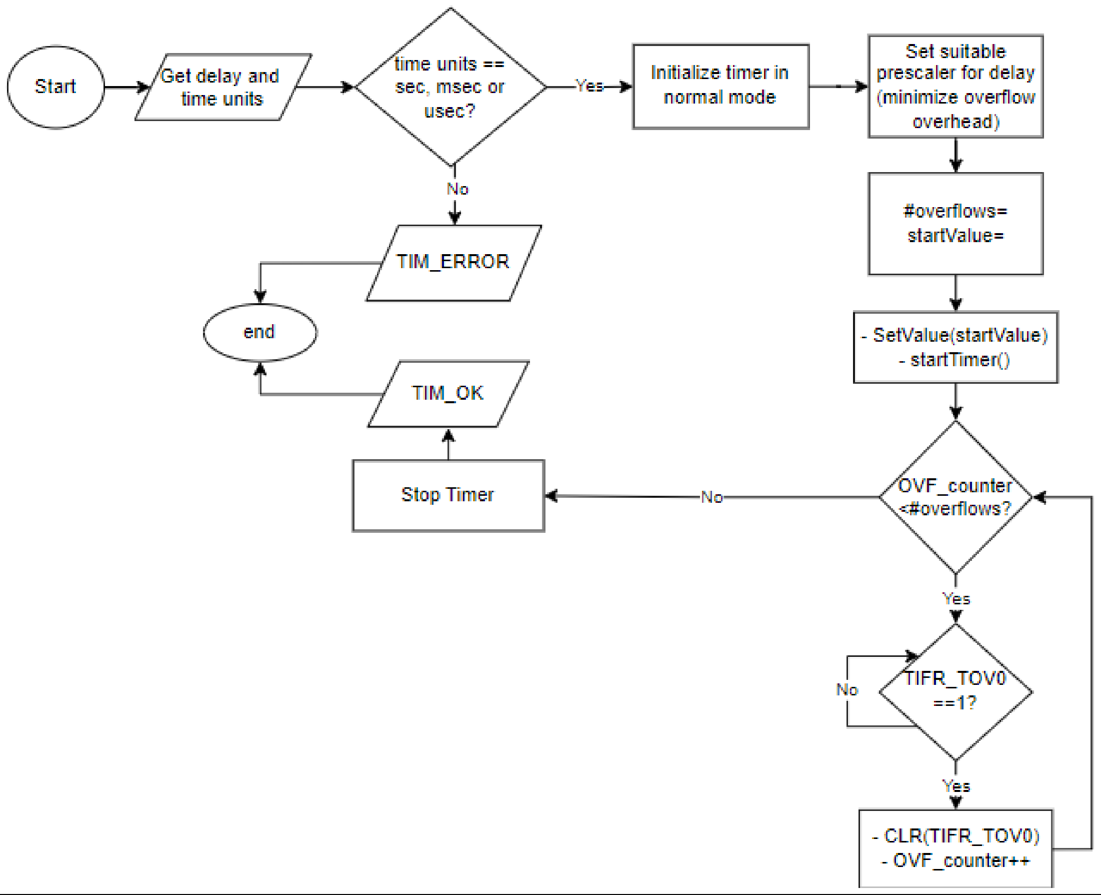
## TIM0_Start


Start
Get prescaler
no_prescaler <prescaler<= External Rising Edge
Yes
Clear timer0 prescaler bits
No
TIM_ERROR
end
TIM_OK
Set timer0 prescaler bits to configured value

## TIM0_Stop

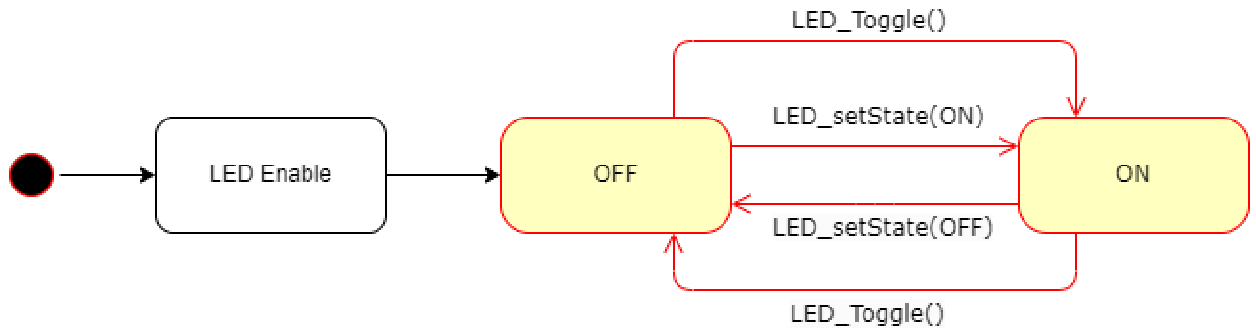


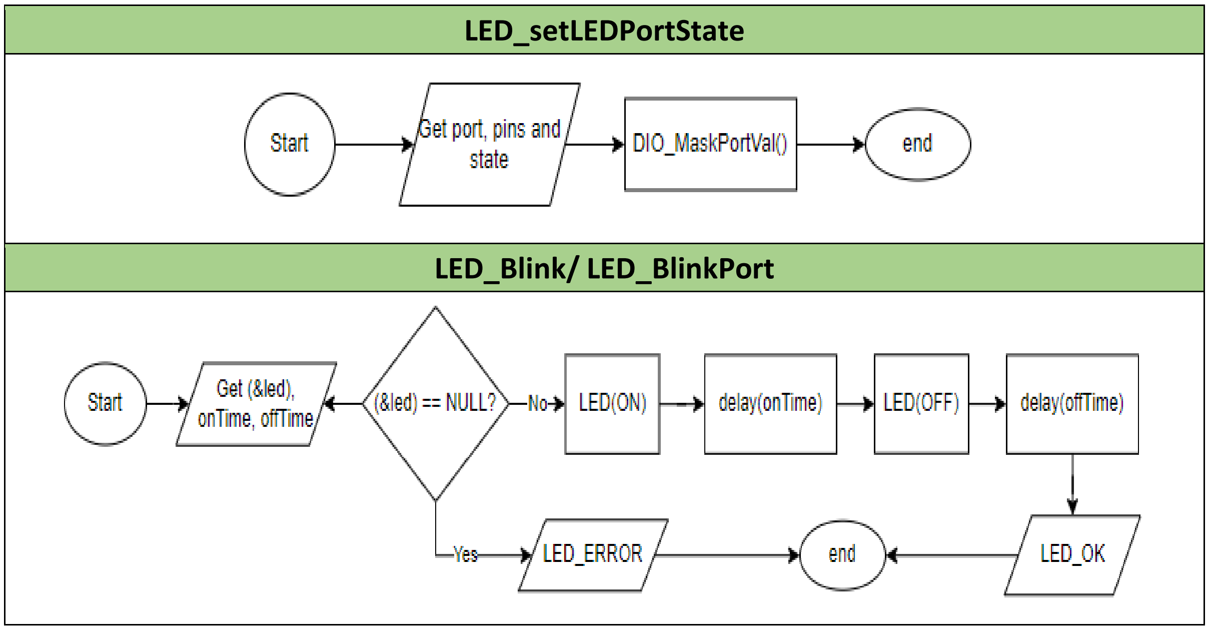
## TIM0_Delay


Start
Get delay and time units
time units == sec, msec or usec?
Yes
Initialize timer in normal mode
Set suitable prescaler for delay (minimize overflow overhead)
No
TIM_ERROR
#overflows= startValue=
end
- SetValue(startValue) - startTimer()
TIM_OK
Stop Timer
No
OVF_counter <#overflows?
Yes
No
TIFR_TOV0 ==1?
Yes
- CLR(TIFR_TOV0) - OVF_counter++

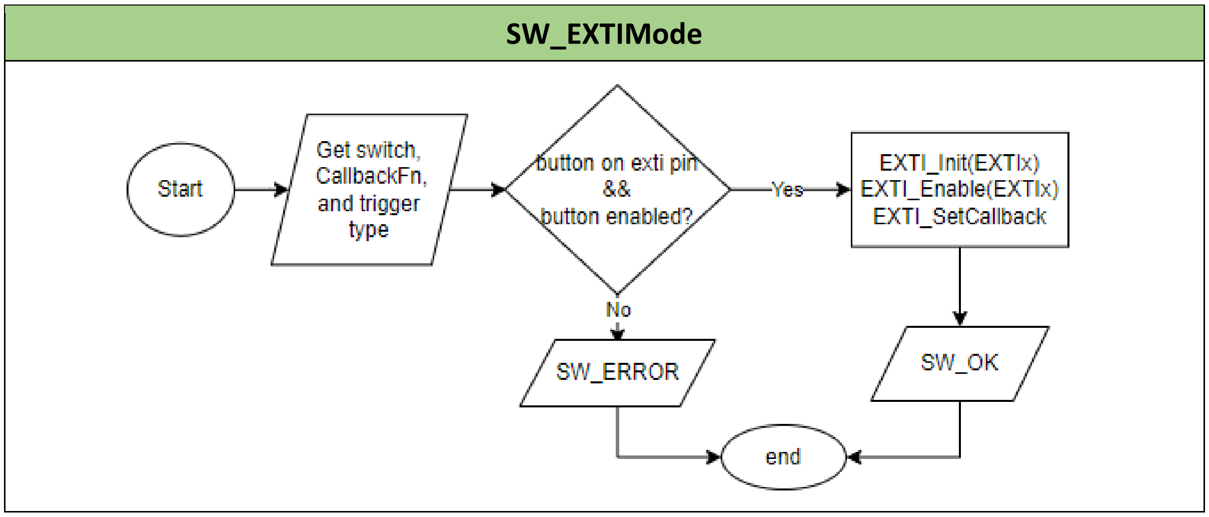
## HAL

- **LED API State Machine**



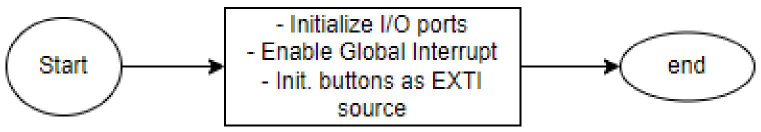


- **Button API Flow Chart**

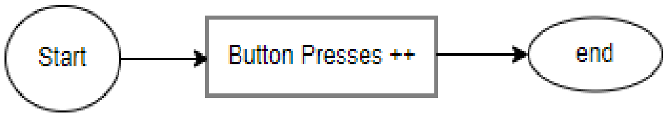

SW_EXTIMode
Start
Get switch, CallbackFn, and trigger type
button on exti pin && button enabled?
Yes
No
EXTI_Init(EXTIx) EXTI_Enable(EXTIx) EXTI_SetCallback
SW_ERROR
SW_OK
end

## APPLICATION

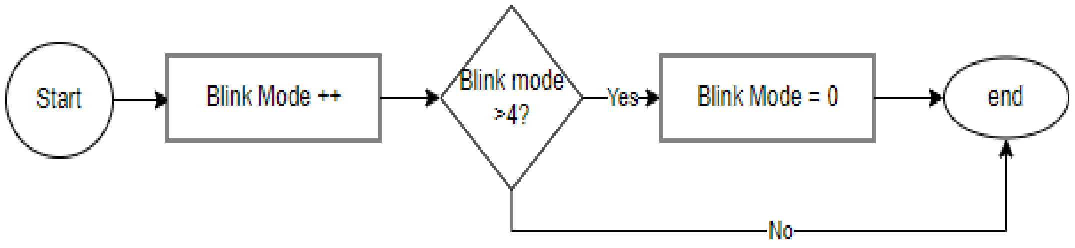
### App_Init



### Button0_Callback



### Button1_Callback


Start
Blink Mode ++
Blink mode >4?
Yes
Blink Mode = 0
end
No