

## Table of Contents

System Description .....	2
System Design .....	3
API .....	4
System Flow Chart .....	9
System Constraints .....	10

## System Description

- Traffic lights are signaling devices positioned at road intersections, pedestrian crossings, and other locations to control the flow of traffic.
- Traffic lights normally consist of three signals, transmitting meaning to drivers and riders through colors and symbols including arrows and bicycles.
- The regular traffic light colors are red, yellow, and green arranged vertically or horizontally in that order.
- The system is a Traffic Light system to manage the traffic of cars
- The system has two modes
  - Normal Mode:
    - Cars' LEDs will be changed every five seconds starting from Green, yellow, red, yellow, and green.
    - The Yellow LED will blink for five seconds before moving to Green or Red LEDs.
  - Pedestrian Mode:
    - Change from normal mode to pedestrian mode when the pedestrian button is pressed.
    - If pressed when the cars' Red LED is on, the pedestrian's Green LED and the cars' Red LEDs will be on for five seconds, this means that pedestrians can cross the street while the pedestrian's Green LED is on.
    - If pressed when the cars' Green LED is on or the cars' Yellow LED is blinking, the pedestrian Red LED will be on then both Yellow LEDs start to blink for five seconds, then the cars' Red LED and pedestrian Green LEDs are on for five seconds, this means that pedestrian must wait until the Green LED is on.
    - At the end of the two states, the cars' Red LED will be off and both Yellow LEDs start blinking for 5 seconds and the pedestrian's Green LED is still on.
    - After the five seconds the pedestrian Green LED will be off and both the pedestrian Red LED and the cars' Green LED will be on.
    - Traffic lights signals are going to the normal mode again.

## System Design

### ➤ Hardware Tools

- ATmega32
- 6 LEDs
- 1 Push Button

### ➤ Peripherals

#### ○ DIO

- ( Digital input-output ) To be able to output something on Microcontroller's Pins
- The Pins of the LEDs are Output
- The pin of the Push Button is Input Pull-Up

#### ○ External Interrupt

- The Interrupt is a signal that has a higher priority than the background system.
- The Push Button is connected to INT0 (External Interrupt Line 0)
- The External Interrupt Lin 0 is configured to sense and generate an interrupt on the Falling Edge.
- This Interrupts the current function and changes to the Pedestrian Mode.

#### ○ Timer 0

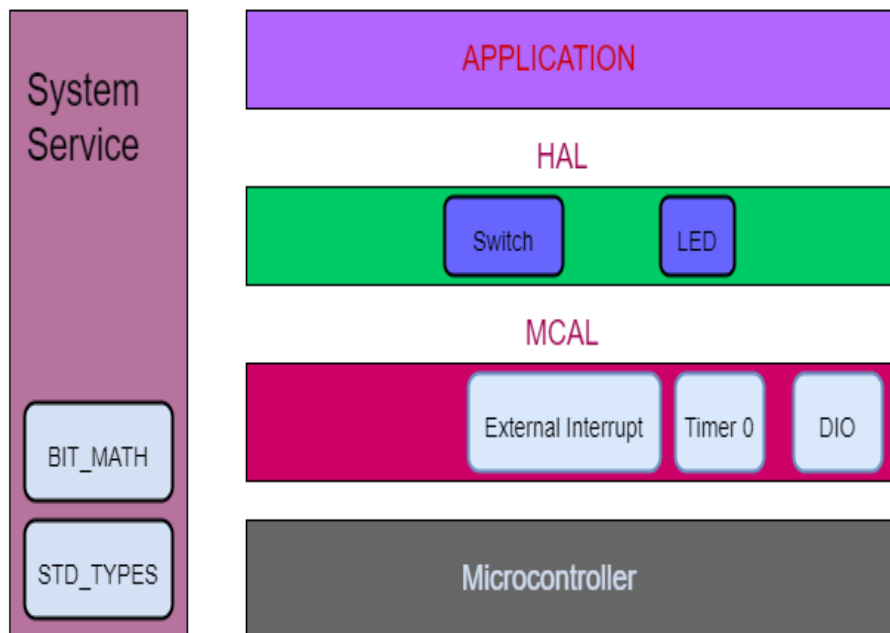
- Timer 0 is used to manage the switching between the possible states of the Traffic Lights
- Every 5 seconds the Timer generates an interrupt and calls the Function that is responsible for changing the states.
- The Timer is Working on the CTC Mode.

### ➤ Software

#### ○ Layered Architecture

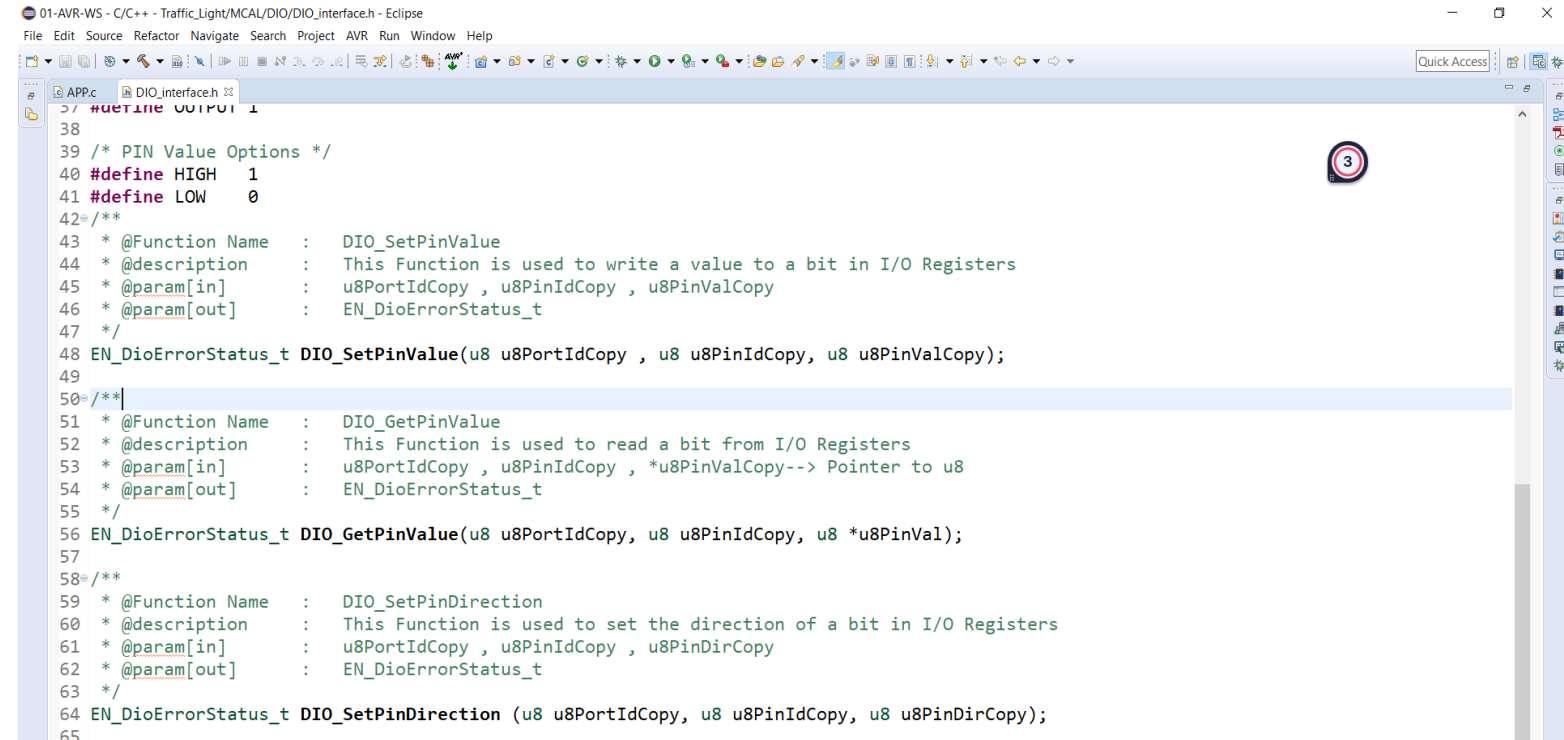
- Hardware:
  - ✓ The Hardware peripherals of the Microcontroller
- MCAL
  - ✓ Microcontroller Abstraction layer
  - ✓ This Contains the peripherals drivers
  - ✓ In each driver we have APIs that directly manage the Hardware registers

- HAL
  - ✓ Hardware Abstraction Layer
  - ✓ Contains the external hardware connected to the Microcontroller (LEDs, Push Button, sensors, ...)
- APPLICATION
  - ✓ Contains the application algorithm of the desired system functionality
- System Service
  - ✓ Contains the standard Types and the Bit math macros



API

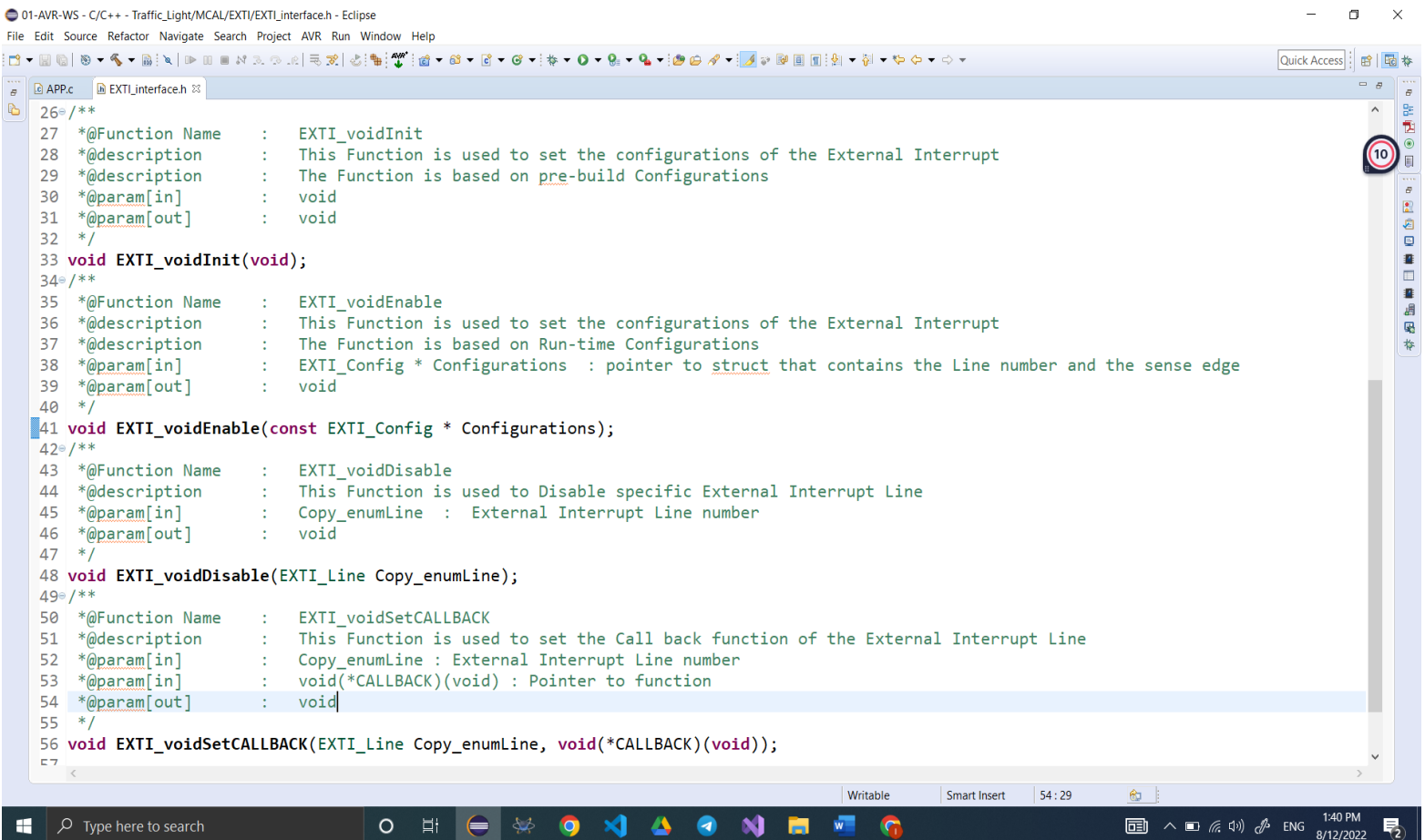
## • DIO



```
01-AVR-WS - C/C++ - Traffic_Light/MCAL/DIO/DIO_interface.h - Eclipse
File Edit Source Refactor Navigate Search Project AVR Run Window Help

APP: C DIO_interface.h
38
39 /* PIN Value Options */
40 #define HIGH 1
41 #define LOW 0
42 /**
43  * @Function Name : DIO_SetPinValue
44  * @description : This Function is used to write a value to a bit in I/O Registers
45  * @param[in] : u8PortIdCopy , u8PinIdCopy , u8PinValCopy
46  * @param[out] : EN_DioErrorStatus_t
47  */
48 EN_DioErrorStatus_t DIO_SetPinValue(u8 u8PortIdCopy , u8 u8PinIdCopy, u8 u8PinValCopy);
49
50 /**
51  * @Function Name : DIO_GetPinValue
52  * @description : This Function is used to read a bit from I/O Registers
53  * @param[in] : u8PortIdCopy , u8PinIdCopy , *u8PinValCopy--> Pointer to u8
54  * @param[out] : EN_DioErrorStatus_t
55  */
56 EN_DioErrorStatus_t DIO_GetPinValue(u8 u8PortIdCopy, u8 u8PinIdCopy, u8 *u8PinVal);
57
58 /**
59  * @Function Name : DIO_SetPinDirection
60  * @description : This Function is used to set the direction of a bit in I/O Registers
61  * @param[in] : u8PortIdCopy , u8PinIdCopy , u8PinDirCopy
62  * @param[out] : EN_DioErrorStatus_t
63  */
64 EN_DioErrorStatus_t DIO_SetPinDirection (u8 u8PortIdCopy, u8 u8PinIdCopy, u8 u8PinDirCopy);
65
```

## • EXTI

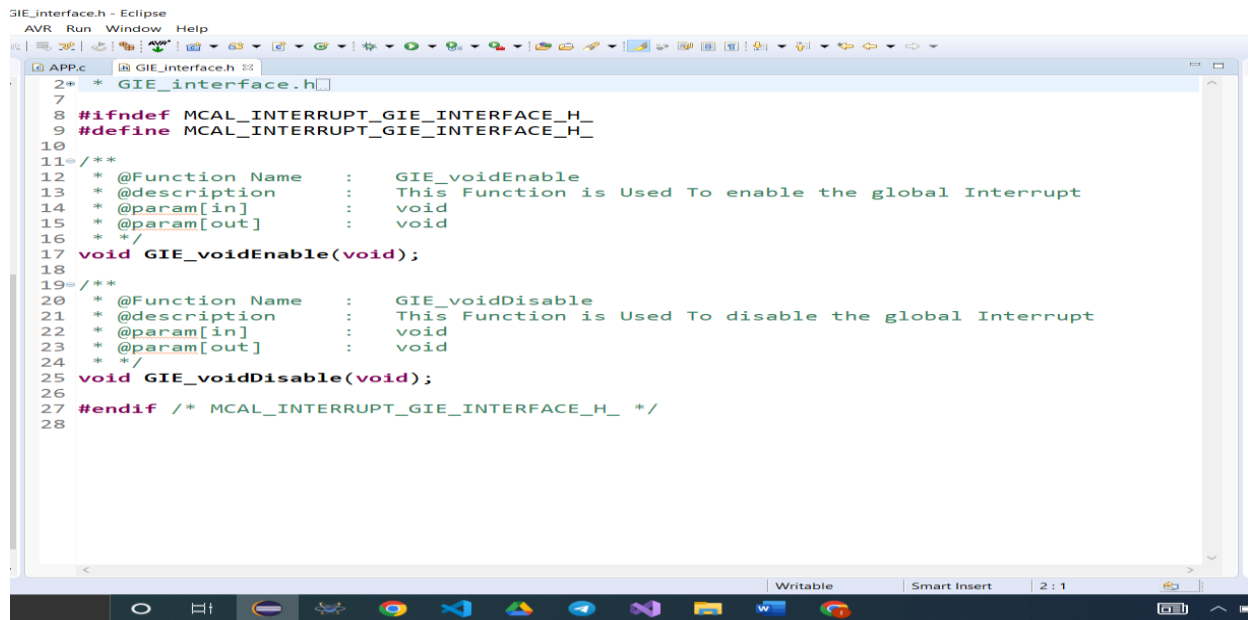


```
01-AVR-WS - C/C++ - Traffic_Light/MCAL/EXTI/EXTI_interface.h - Eclipse
File Edit Source Refactor Navigate Search Project AVR Run Window Help

APP: C EXTI_interface.h
26 /**
27  * @Function Name : EXTI_voidInit
28  * @description : This Function is used to set the configurations of the External Interrupt
29  * @description : The Function is based on pre-build Configurations
30  * @param[in] : void
31  * @param[out] : void
32  */
33 void EXTI_voidInit(void);
34 /**
35  * @Function Name : EXTI_voidEnable
36  * @description : This Function is used to set the configurations of the External Interrupt
37  * @description : The Function is based on Run-time Configurations
38  * @param[in] : EXTI_Config * Configurations : pointer to struct that contains the Line number and the sense edge
39  * @param[out] : void
40  */
41 void EXTI_voidEnable(const EXTI_Config * Configurations);
42 /**
43  * @Function Name : EXTI_voidDisable
44  * @description : This Function is used to Disable specific External Interrupt Line
45  * @param[in] : Copy_enumLine : External Interrupt Line number
46  * @param[out] : void
47  */
48 void EXTI_voidDisable(EXTI_Line Copy_enumLine);
49 /**
50  * @Function Name : EXTI_voidSetCALLBACK
51  * @description : This Function is used to set the Call back function of the External Interrupt Line
52  * @param[in] : Copy_enumLine : External Interrupt Line number
53  * @param[in] : void(*CALLBACK)(void) : Pointer to function
54  * @param[out] : void
55  */
56 void EXTI_voidSetCALLBACK(EXTI_Line Copy_enumLine, void(*CALLBACK)(void));
57
```

Writabe Smart Insert 54:29 ENG 1:40 PM 8/12/2022

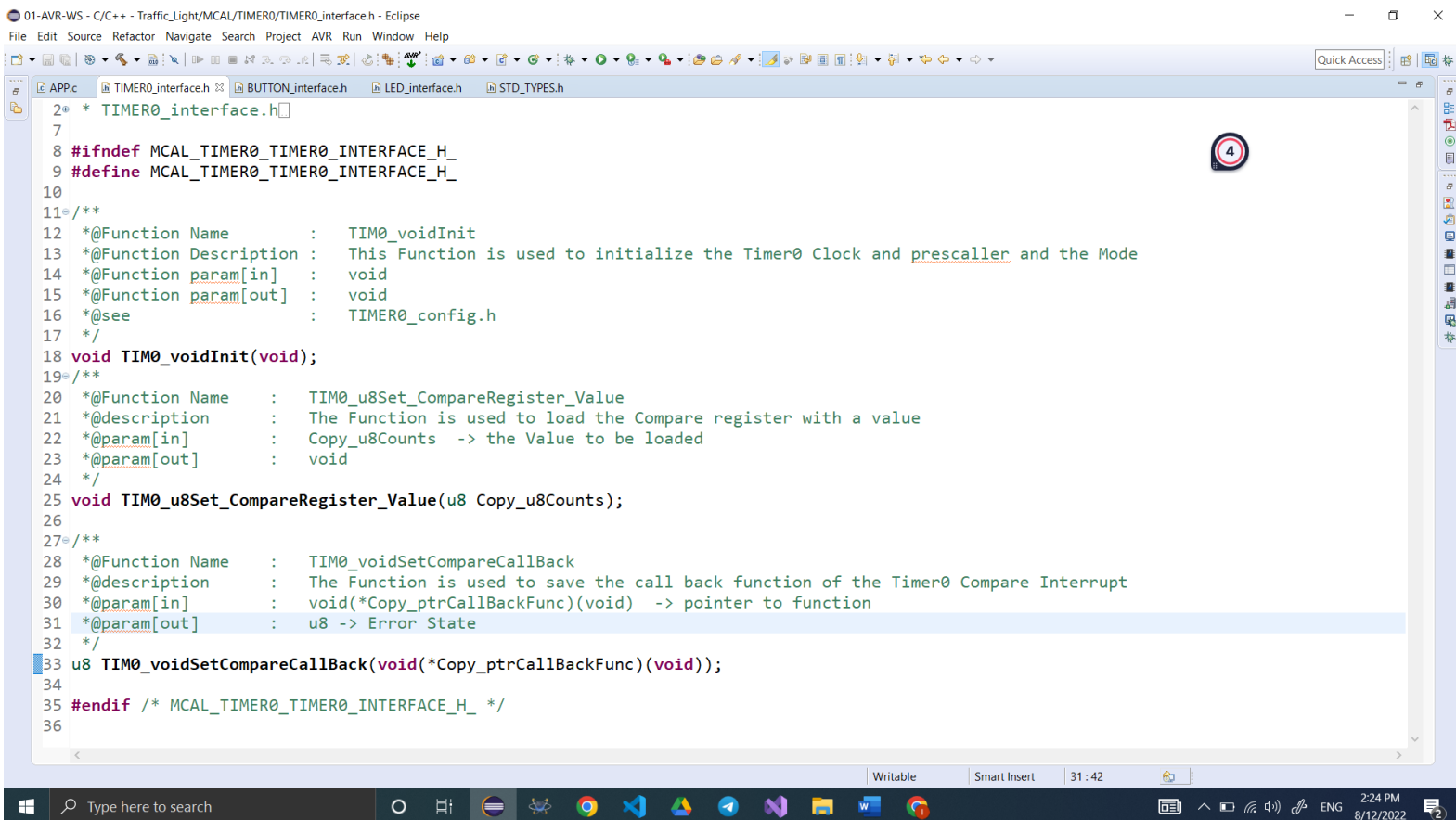
- Global Interrupt



The screenshot shows the Eclipse IDE with the file GIE\_interface.h open. The code defines two functions: GIE\_voidEnable and GIE\_voidDisable. Both functions are marked with @Function Name, @description, @param[in], and @param[out] tags. The GIE\_voidEnable function is described as 'This Function is Used To enable the global Interrupt' and the GIE\_voidDisable function is described as 'This Function is Used To disable the global Interrupt'. The code is enclosed in an #ifndef...#endif block with the identifier MCAL\_INTERRUPT\_GIE\_INTERFACE\_H\_.

```
2* * GIE_interface.h
7
8 #ifndef MCAL_INTERRUPT_GIE_INTERFACE_H_
9 #define MCAL_INTERRUPT_GIE_INTERFACE_H_
10
11 /**
12  * @Function Name      :   GIE_voidEnable
13  * @description        :   This Function is Used To enable the global Interrupt
14  * @param[in]          :   void
15  * @param[out]         :   void
16  */
17 void GIE_voidEnable(void);
18
19 /**
20  * @Function Name      :   GIE_voidDisable
21  * @description        :   This Function is Used To disable the global Interrupt
22  * @param[in]          :   void
23  * @param[out]         :   void
24  */
25 void GIE_voidDisable(void);
26
27 #endif /* MCAL_INTERRUPT_GIE_INTERFACE_H_ */
28
```

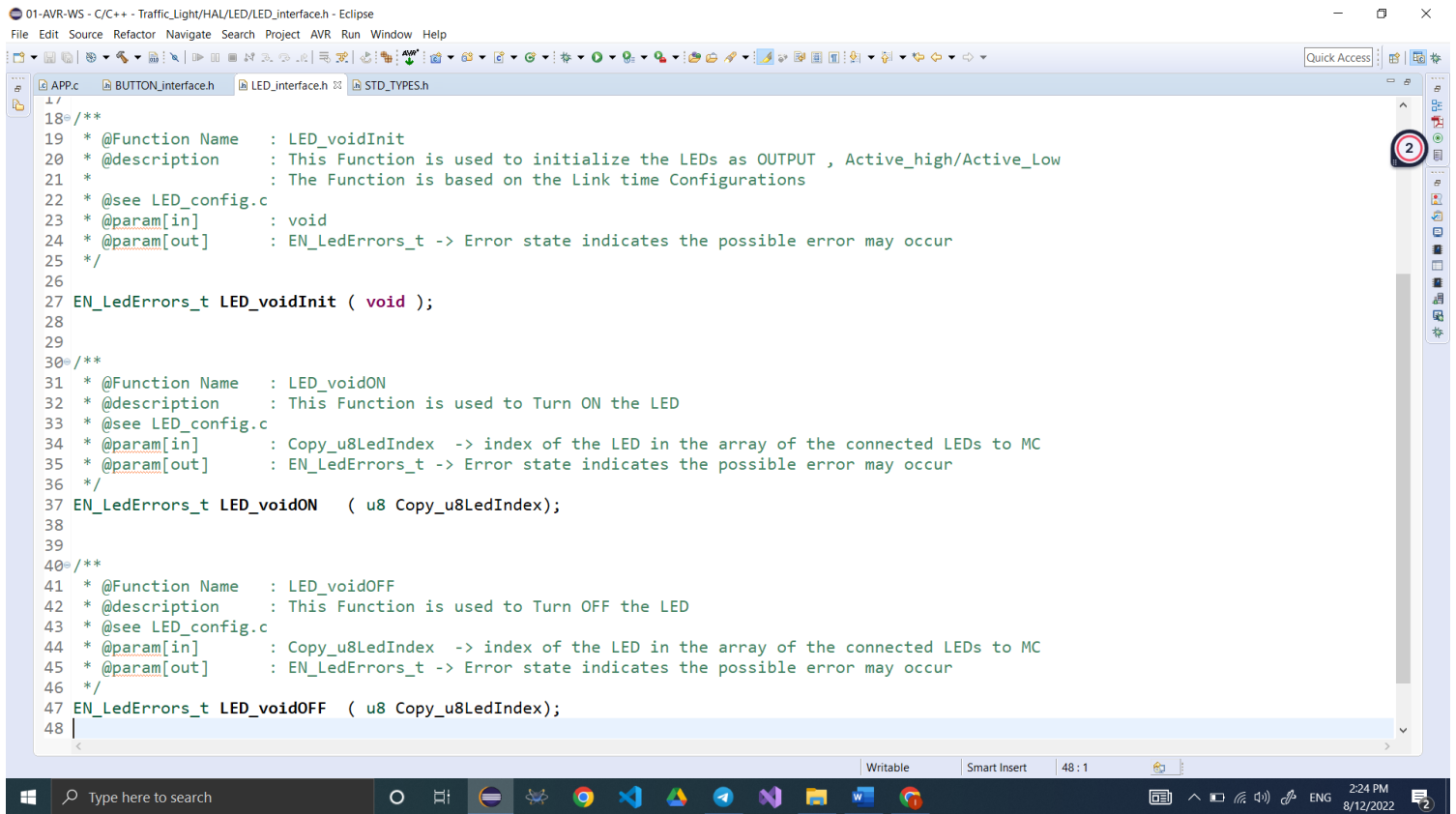
- Timer



The screenshot shows the Eclipse IDE with the file TIMER0\_interface.h open. The code defines three functions: TIM0\_voidInit, TIM0\_u8Set\_CompareRegister\_Value, and TIM0\_voidSetCompareCallBack. The TIM0\_voidInit function is described as 'This Function is used to initialize the Timer0 Clock and prescaler and the Mode'. The TIM0\_u8Set\_CompareRegister\_Value function is described as 'The Function is used to load the Compare register with a value'. The TIM0\_voidSetCompareCallBack function is described as 'The Function is used to save the call back function of the Timer0 Compare Interrupt'. The code is enclosed in an #ifndef...#endif block with the identifier MCAL\_TIMER0\_TIMER0\_INTERFACE\_H\_.

```
2* * TIMER0_interface.h
7
8 #ifndef MCAL_TIMER0_TIMER0_INTERFACE_H_
9 #define MCAL_TIMER0_TIMER0_INTERFACE_H_
10
11 /**
12  * @Function Name      :   TIM0_voidInit
13  * @Function Description :   This Function is used to initialize the Timer0 Clock and prescaler and the Mode
14  * @Function param[in]  :   void
15  * @Function param[out] :   void
16  * @see                :   TIMER0_config.h
17  */
18 void TIM0_voidInit(void);
19 /**
20  * @Function Name      :   TIM0_u8Set_CompareRegister_Value
21  * @description        :   The Function is used to load the Compare register with a value
22  * @param[in]          :   Copy_u8Counts -> the Value to be loaded
23  * @param[out]         :   void
24  */
25 void TIM0_u8Set_CompareRegister_Value(u8 Copy_u8Counts);
26
27 /**
28  * @Function Name      :   TIM0_voidSetCompareCallBack
29  * @description        :   The Function is used to save the call back function of the Timer0 Compare Interrupt
30  * @param[in]          :   void(*Copy_ptrCallBackFunc)(void) -> pointer to function
31  * @param[out]         :   u8 -> Error State
32  */
33 u8 TIM0_voidSetCompareCallBack(void(*Copy_ptrCallBackFunc)(void));
34
35 #endif /* MCAL_TIMER0_TIMER0_INTERFACE_H_ */
36
```

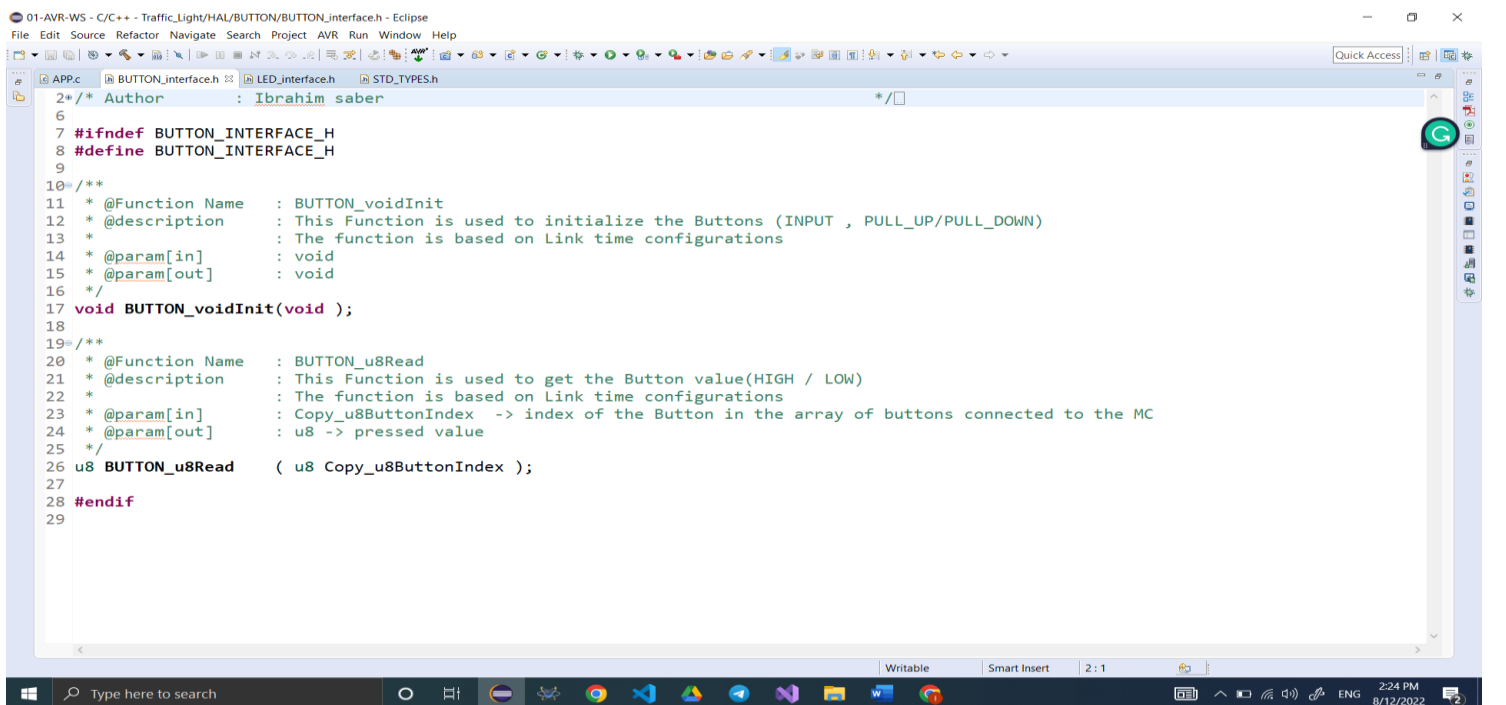
## • LED



The screenshot shows the Eclipse IDE with the file LED\_interface.h open. The code defines three functions: LED\_voidInit, LED\_voidON, and LED\_voidOFF. Each function has a detailed comment block describing its purpose and parameters. The LED\_voidInit function initializes LEDs as OUTPUT, Active\_high/Active\_Low. LED\_voidON turns ON the LED, and LED\_voidOFF turns OFF the LED. All functions take a Copy\_u8LedIndex parameter and return an EN\_LedErrors\_t value.

```
18 /**
19  * @Function Name : LED_voidInit
20  * @description : This Function is used to initialize the LEDs as OUTPUT , Active_high/Active_Low
21  * : The Function is based on the Link time Configurations
22  * @see LED_config.c
23  * @param[in] : void
24  * @param[out] : EN_LedErrors_t -> Error state indicates the possible error may occur
25  */
26
27 EN_LedErrors_t LED_voidInit ( void );
28
29
30 /**
31  * @Function Name : LED_voidON
32  * @description : This Function is used to Turn ON the LED
33  * @see LED_config.c
34  * @param[in] : Copy_u8LedIndex -> index of the LED in the array of the connected LEDs to MC
35  * @param[out] : EN_LedErrors_t -> Error state indicates the possible error may occur
36  */
37 EN_LedErrors_t LED_voidON ( u8 Copy_u8LedIndex);
38
39
40 /**
41  * @Function Name : LED_voidOFF
42  * @description : This Function is used to Turn OFF the LED
43  * @see LED_config.c
44  * @param[in] : Copy_u8LedIndex -> index of the LED in the array of the connected LEDs to MC
45  * @param[out] : EN_LedErrors_t -> Error state indicates the possible error may occur
46  */
47 EN_LedErrors_t LED_voidOFF ( u8 Copy_u8LedIndex);
48
```

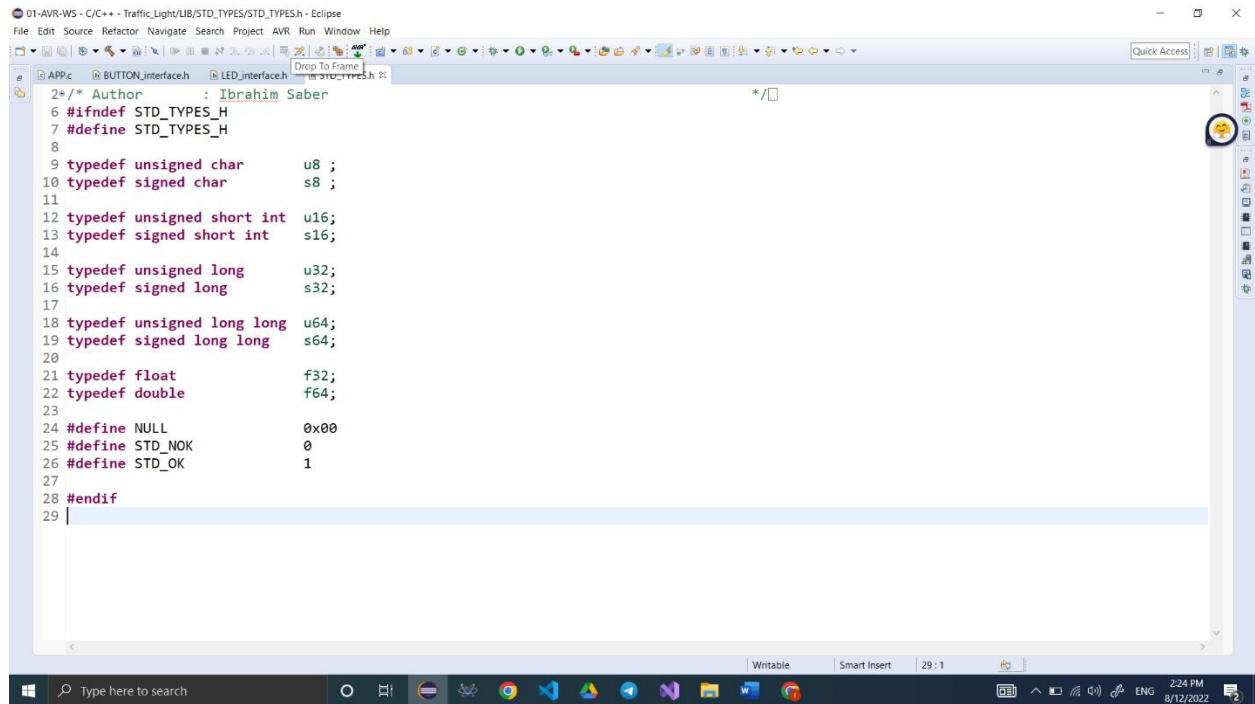
## Push Button



The screenshot shows the Eclipse IDE with the file BUTTON\_interface.h open. The code defines two functions: BUTTON\_voidInit and BUTTON\_u8Read. BUTTON\_voidInit initializes buttons as INPUT, PULL\_UP/PULL\_DOWN. BUTTON\_u8Read gets the Button value (HIGH / LOW). Both functions take a Copy\_u8ButtonIndex parameter and return a u8 value. The code is enclosed in a #ifndef and #define block.

```
2 * // Author : Ibrahim saber */
6
7 #ifndef BUTTON_INTERFACE_H
8 #define BUTTON_INTERFACE_H
9
10 /**
11  * @Function Name : BUTTON_voidInit
12  * @description : This Function is used to initialize the Buttons (INPUT , PULL_UP/PULL_DOWN)
13  * : The function is based on Link time configurations
14  * @param[in] : void
15  * @param[out] : void
16  */
17 void BUTTON_voidInit(void );
18
19 /**
20  * @Function Name : BUTTON_u8Read
21  * @description : This Function is used to get the Button value(HIGH / LOW)
22  * : The function is based on Link time configurations
23  * @param[in] : Copy_u8ButtonIndex -> index of the Button in the array of buttons connected to the MC
24  * @param[out] : u8 -> pressed value
25  */
26 u8 BUTTON_u8Read ( u8 Copy_u8ButtonIndex );
27
28 #endif
29
```

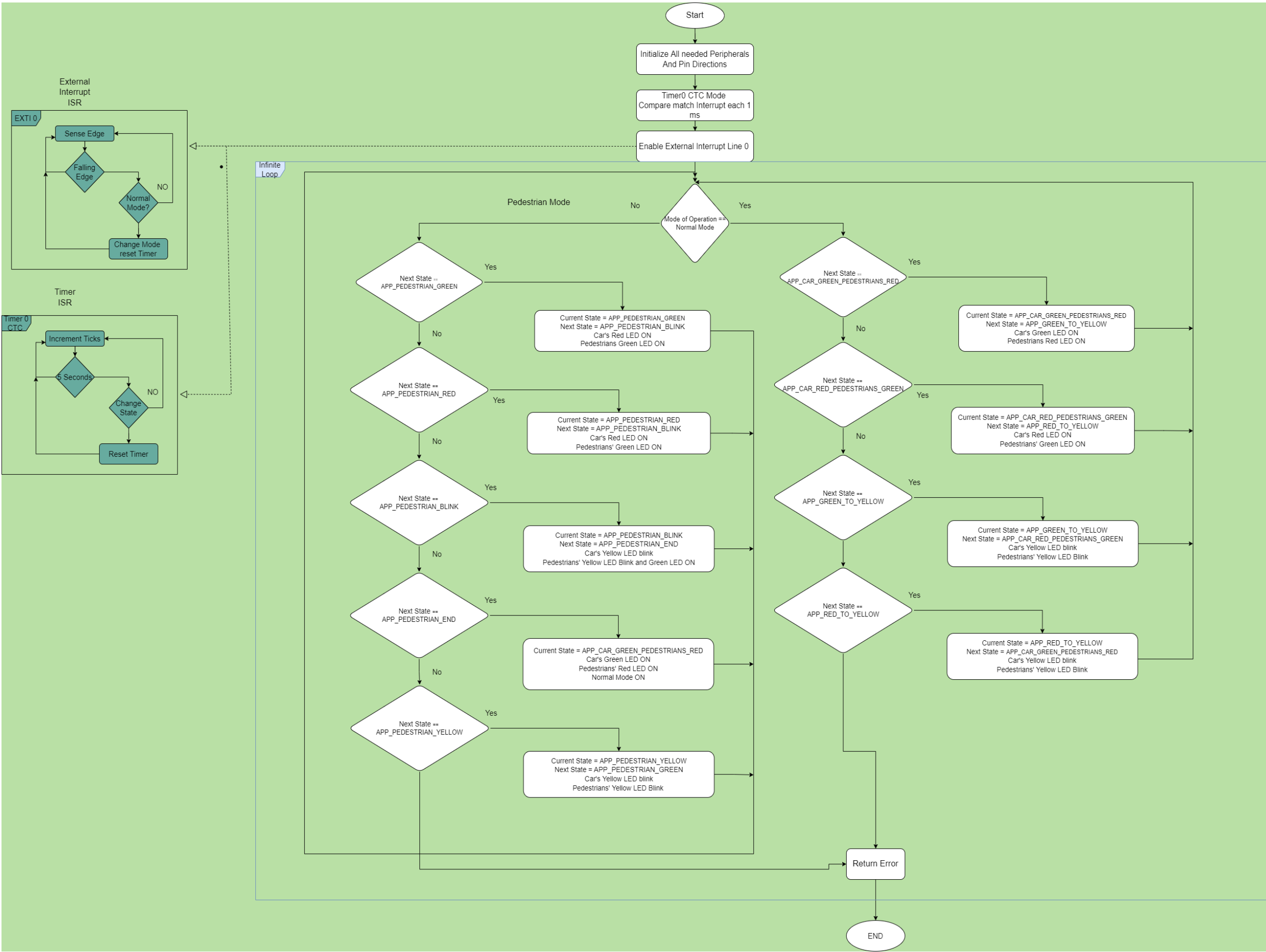
# Standard Types



```
2 /* Author : Ibrahim Saber */
6 #ifndef STD_TYPES_H
7 #define STD_TYPES_H
8
9 typedef unsigned char u8 ;
10 typedef signed char s8 ;
11
12 typedef unsigned short int u16;
13 typedef signed short int s16;
14
15 typedef unsigned long u32;
16 typedef signed long s32;
17
18 typedef unsigned long long u64;
19 typedef signed long long s64;
20
21 typedef float f32;
22 typedef double f64;
23
24 #define NULL 0x00
25 #define STD_NOK 0
26 #define STD_OK 1
27
28 #endif
29 |
```



System Flow Chart



## System Constraints

- The system is done according to the description and the instruction video