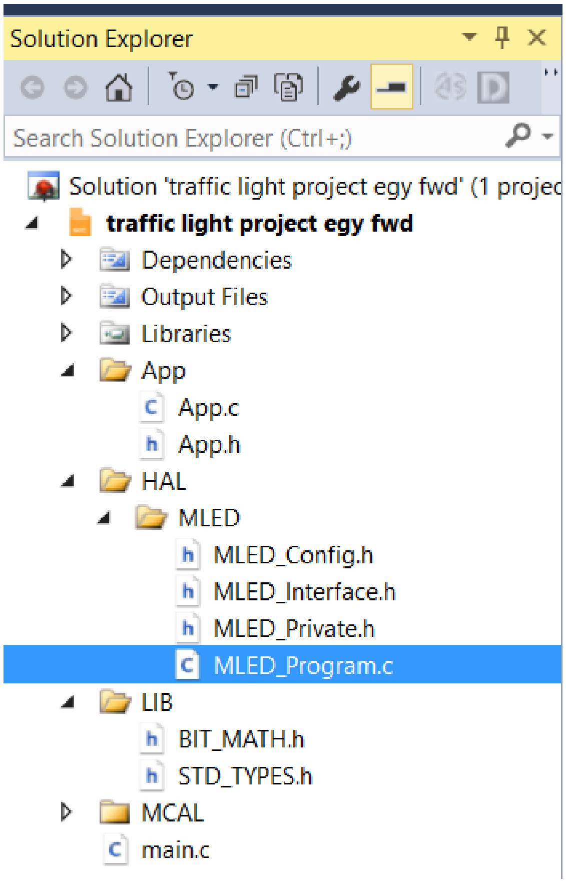
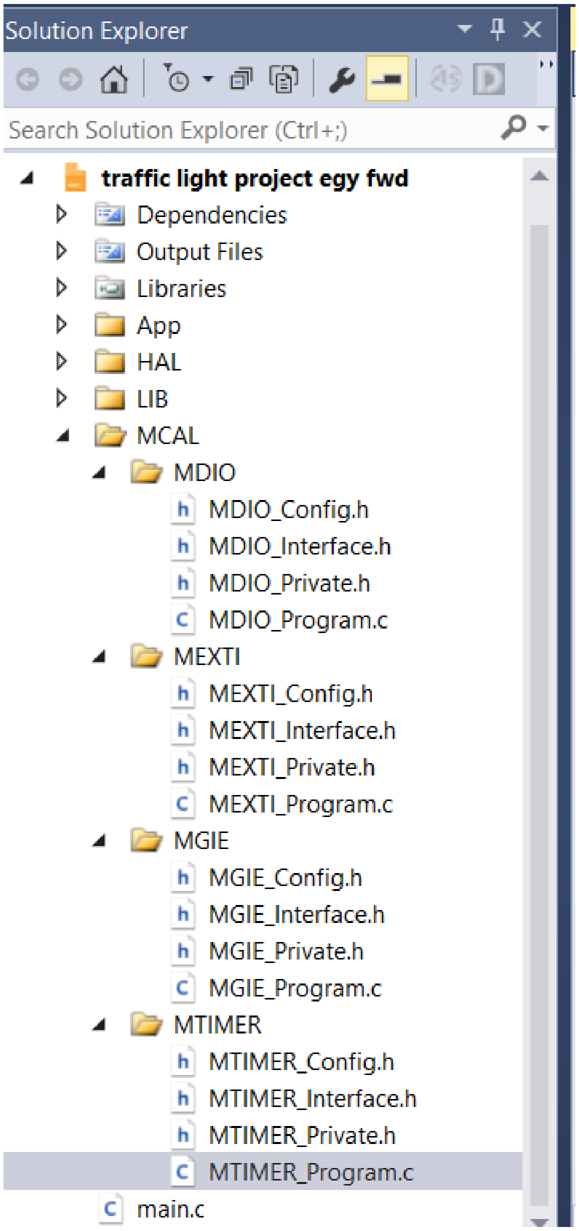
Solution explorer

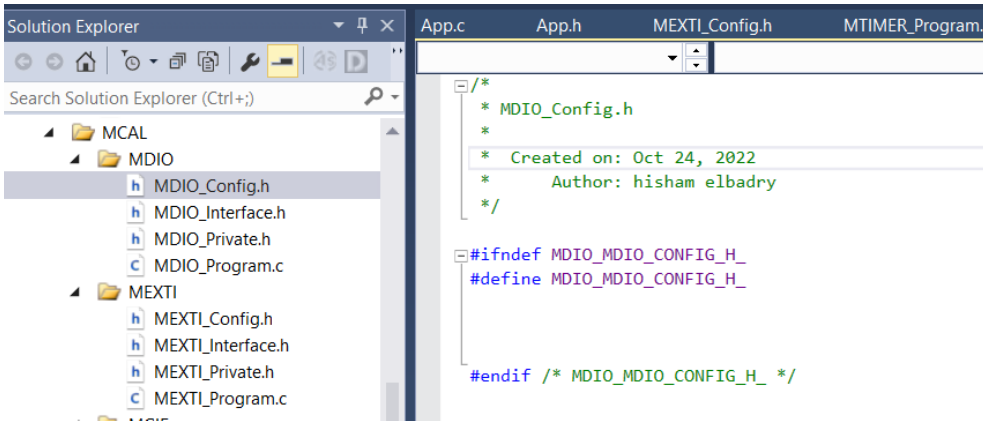


- Solution 'traffic light project egy fwd' (1 projec
  - traffic light project egy fwd
    - Dependencies
    - Output Files
    - Libraries
    - App
      - App.c
      - App.h
    - HAL
      - MLED
        - MLED_Config.h
        - MLED_Interface.h
        - MLED_Private.h
        - MLED_Program.c
    - LIB
      - BIT_MATH.h
      - STD_TYPES.h
    - MCAL
    - main.c

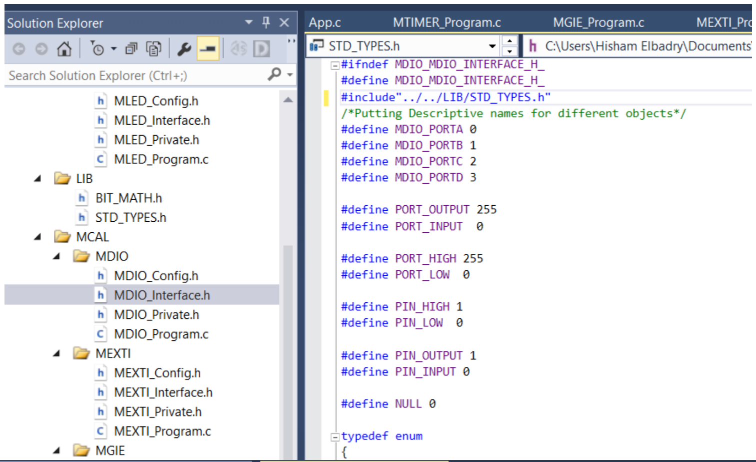


- traffic light project egy fwd
  - Dependencies
  - Output Files
  - Libraries
  - App
  - HAL
  - LIB
  - MCAL
    - MDIO
      - MDIO_Config.h
      - MDIO_Interface.h
      - MDIO_Private.h
      - MDIO_Program.c
    - MEXTI
      - MEXTI_Config.h
      - MEXTI_Interface.h
      - MEXTI_Private.h
      - MEXTI_Program.c
    - MGIE
      - MGIE_Config.h
      - MGIE_Interface.h
      - MGIE_Private.h
      - MGIE_Program.c
    - MTIMER
      - MTIMER_Config.h
      - MTIMER_Interface.h
      - MTIMER_Private.h
      - MTIMER_Program.c
  - main.c

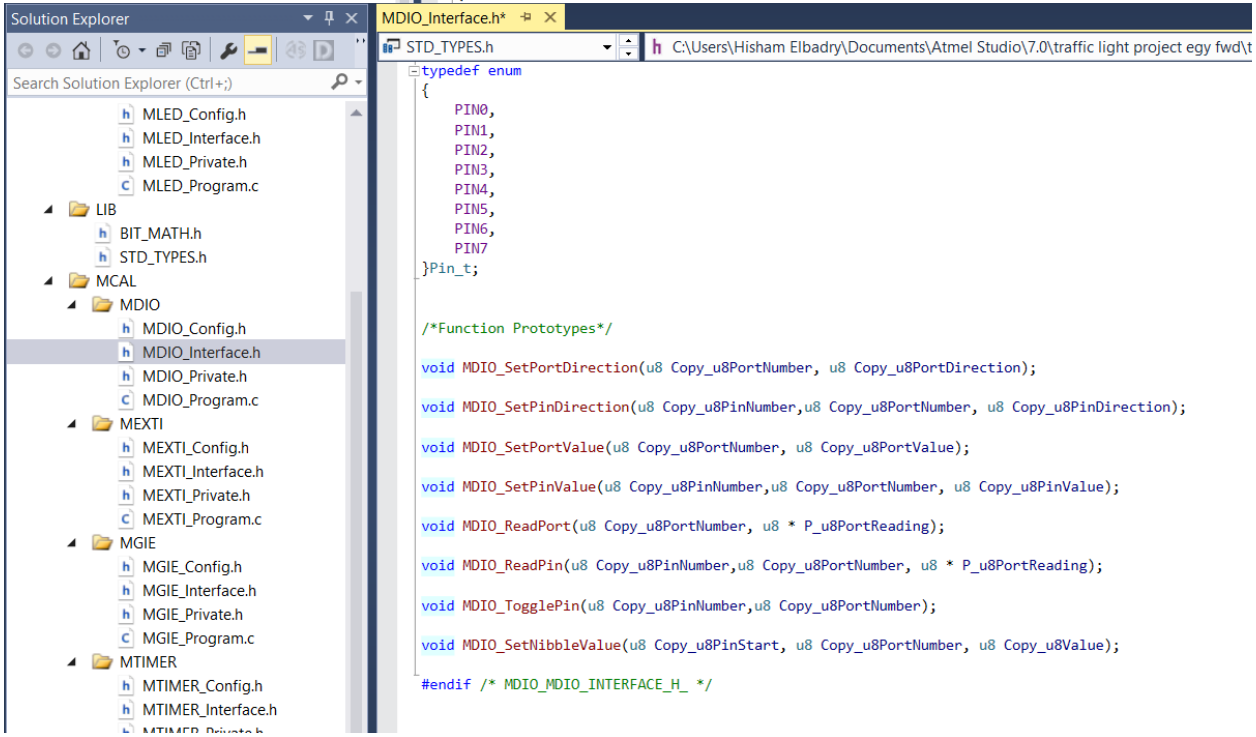
DIO files



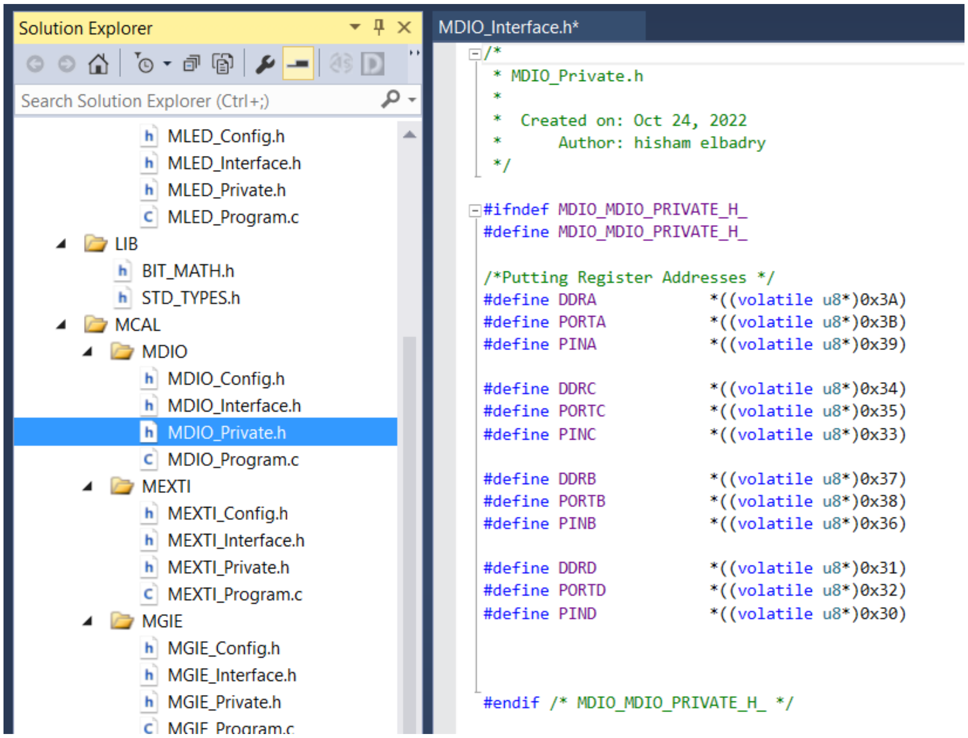
```
/*
 * MDIO_Config.h
 *
 *  Created on: Oct 24, 2022
 *      Author: hisham elbadry
 */

#ifndef MDIO_MDIO_CONFIG_H_
#define MDIO_MDIO_CONFIG_H_



#endif /* MDIO_MDIO_CONFIG_H_ */
```



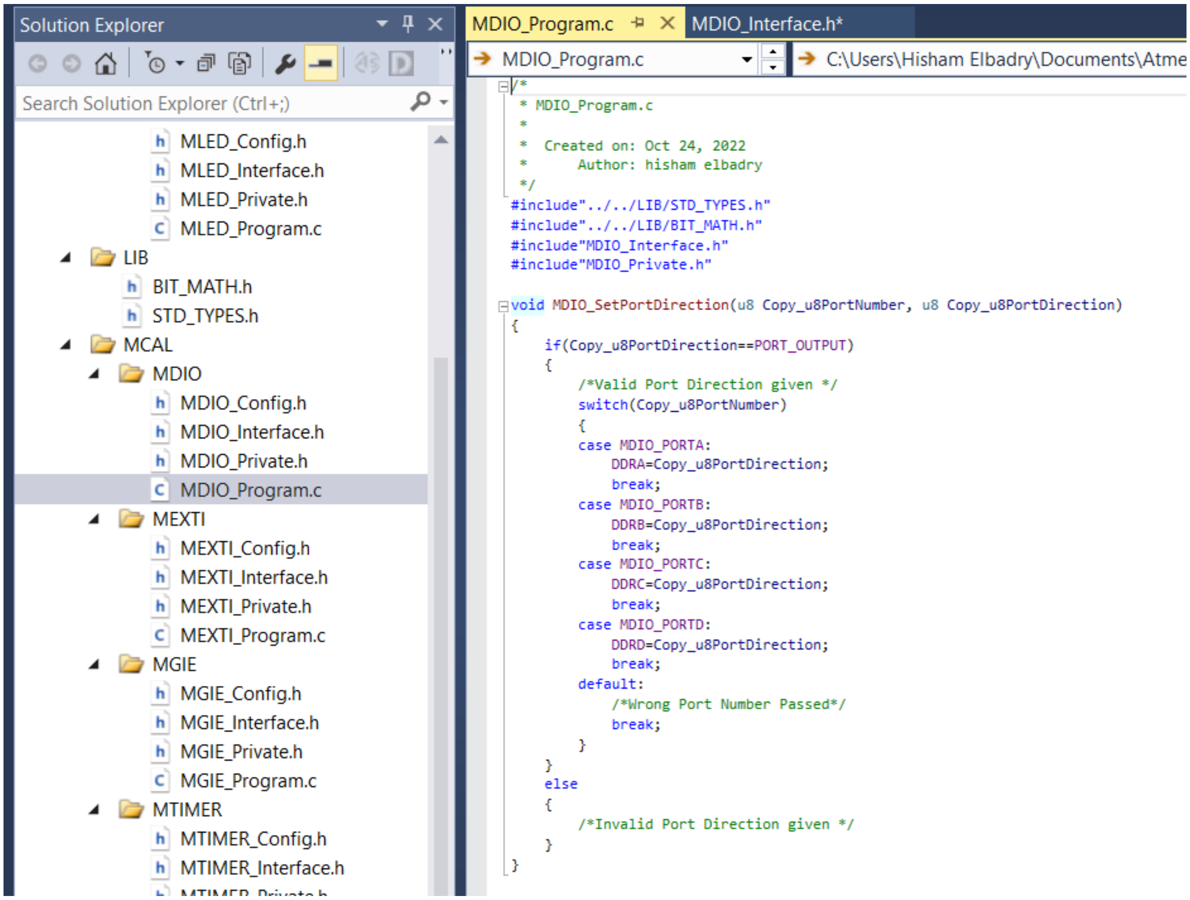
```
#ifndef MDIO_MDIO_INTERFACE_H_
#define MDIO_MDIO_INTERFACE_H_
#include"../../LIB/STD_TYPES.h"
/*Putting Descriptive names for different objects*/
#define MDIO_PORTA 0
#define MDIO_PORTB 1
#define MDIO_PORTC 2
#define MDIO_PORTD 3

#define PORT_OUTPUT 255
#define PORT_INPUT  0

#define PORT_HIGH 255
#define PORT_LOW  0

#define PIN_HIGH 1
#define PIN_LOW  0

#define PIN_OUTPUT 1
#define PIN_INPUT 0

#define NULL 0

typedef enum
{
```



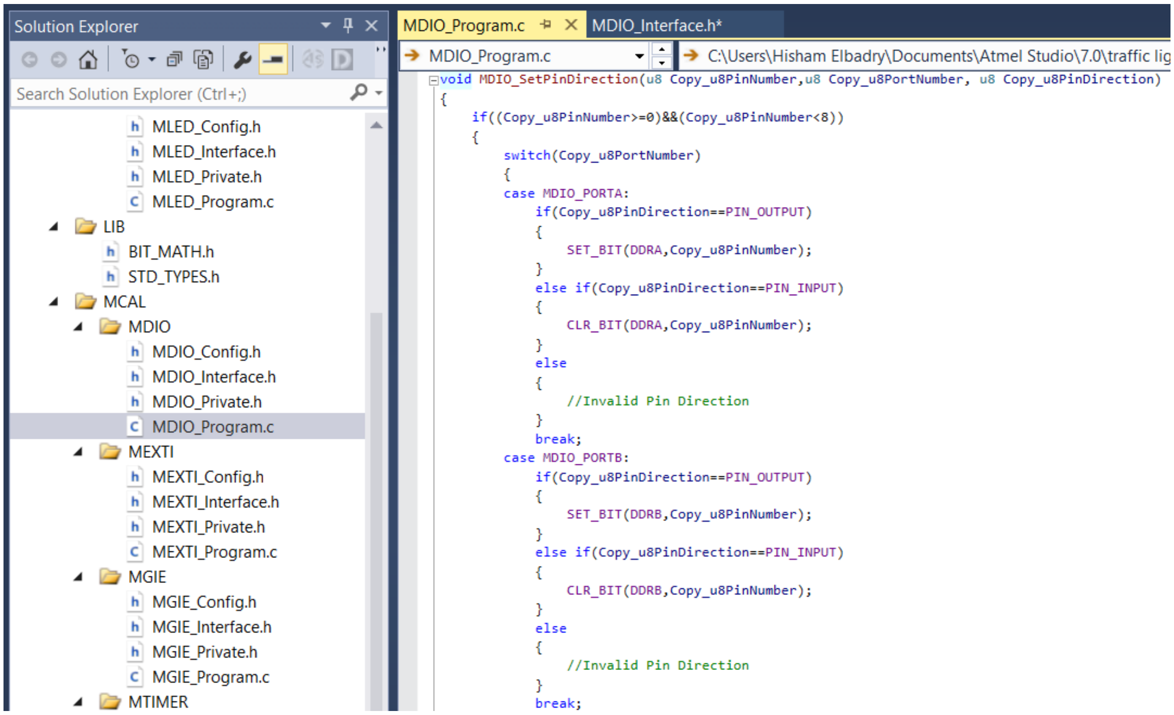
```
typedef enum
{
    PIN0,
    PIN1,
    PIN2,
    PIN3,
    PIN4,
    PIN5,
    PIN6,
    PIN7
}Pin_t;


/*Function Prototypes*/

void MDIO_SetPortDirection(u8 Copy_u8PortNumber, u8 Copy_u8PortDirection);

void MDIO_SetPinDirection(u8 Copy_u8PinNumber,u8 Copy_u8PortNumber, u8 Copy_u8PinDirection);

void MDIO_SetPortValue(u8 Copy_u8PortNumber, u8 Copy_u8PortValue);

void MDIO_SetPinValue(u8 Copy_u8PinNumber,u8 Copy_u8PortNumber, u8 Copy_u8PinValue);

void MDIO_ReadPort(u8 Copy_u8PortNumber, u8 * P_u8PortReading);

void MDIO_ReadPin(u8 Copy_u8PinNumber,u8 Copy_u8PortNumber, u8 * P_u8PortReading);

void MDIO_TogglePin(u8 Copy_u8PinNumber,u8 Copy_u8PortNumber);

void MDIO_SetNibbleValue(u8 Copy_u8PinStart, u8 Copy_u8PortNumber, u8 Copy_u8Value);

#endif /* MDIO_MDIO_INTERFACE_H_ */
```



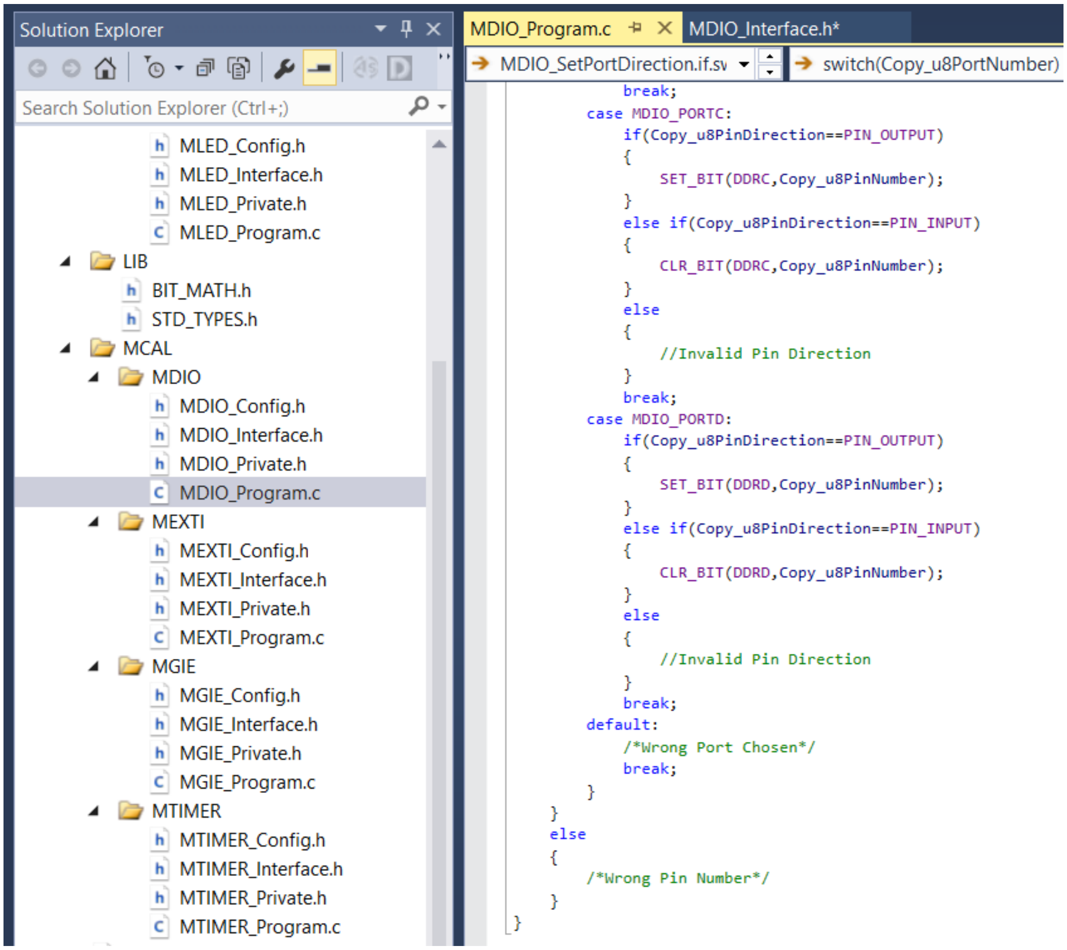
```
/*
 * MDIO_Private.h
 *
 *  Created on: Oct 24, 2022
 *      Author: hisham elbadry
 */

#ifndef MDIO_MDIO_PRIVATE_H_
#define MDIO_MDIO_PRIVATE_H_

/*Putting Register Addresses */
#define DDRA            *((volatile u8*)0x3A)
#define PORTA           *((volatile u8*)0x3B)
#define PINA            *((volatile u8*)0x39)

#define DDRC            *((volatile u8*)0x34)
#define PORTC           *((volatile u8*)0x35)
#define PINC            *((volatile u8*)0x33)

#define DDRB            *((volatile u8*)0x37)
#define PORTB           *((volatile u8*)0x38)
#define PINB            *((volatile u8*)0x36)

#define DDRD            *((volatile u8*)0x31)
#define PORTD           *((volatile u8*)0x32)
#define PIND            *((volatile u8*)0x30)



#endif /* MDIO_MDIO_PRIVATE_H_ */
```



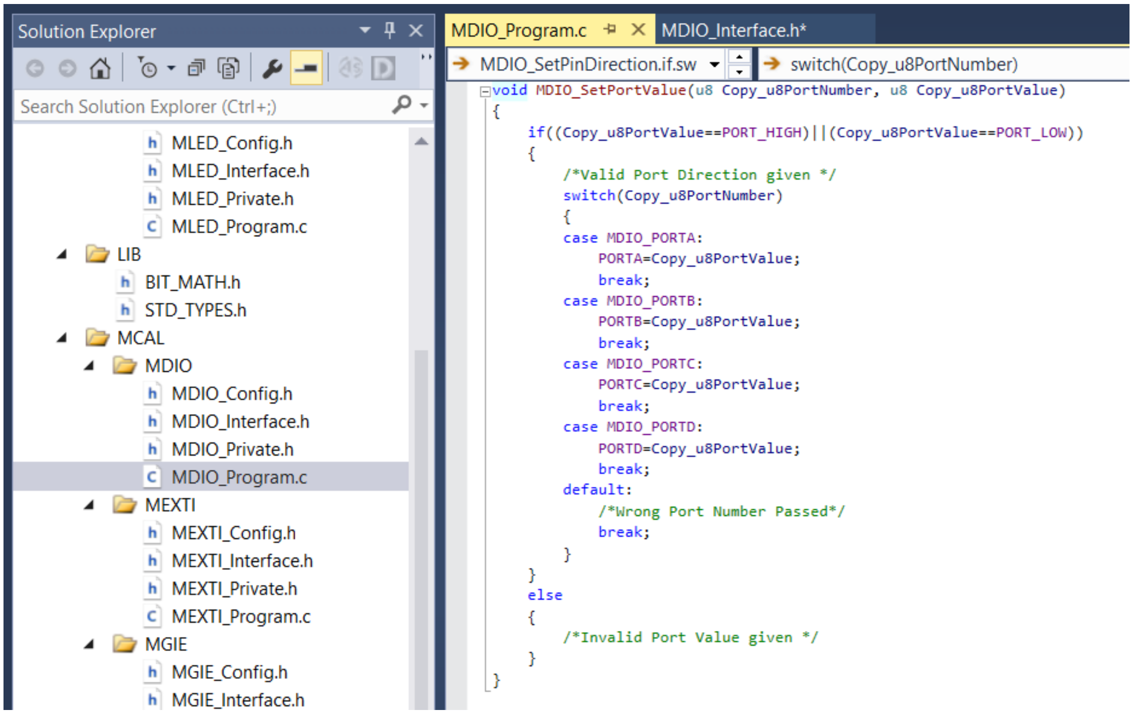
```
/*
 * MDIO_Program.c
 *
 *  Created on: Oct 24, 2022
 *      Author: hisham elbadry
 */
#include"../../LIB/STD_TYPES.h"
#include"../../LIB/BIT_MATH.h"
#include"MDIO_Interface.h"
#include"MDIO_Private.h"

void MDIO_SetPortDirection(u8 Copy_u8PortNumber, u8 Copy_u8PortDirection)
{
    if(Copy_u8PortDirection==PORT_OUTPUT)
    {
        /*Valid Port Direction given */
        switch(Copy_u8PortNumber)
        {
        case MDIO_PORTA:
            DDRA=Copy_u8PortDirection;
            break;
        case MDIO_PORTB:
            DDRB=Copy_u8PortDirection;
            break;
        case MDIO_PORTC:
            DDRC=Copy_u8PortDirection;
            break;
        case MDIO_PORTD:
            DDRD=Copy_u8PortDirection;
            break;
        default:
            /*Wrong Port Number Passed*/
            break;
        }
    }
    else
    {
        /*Invalid Port Direction given */
    }
}
```



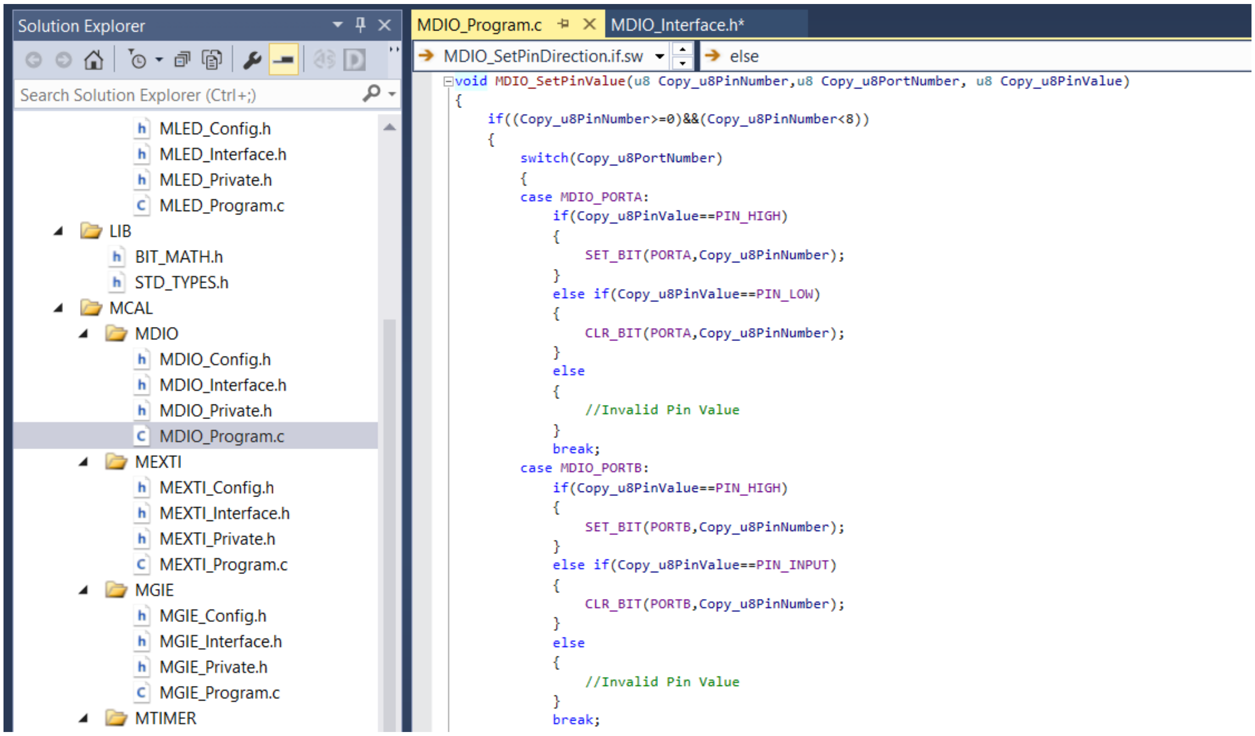
```
void MDIO_SetPinDirection(u8 Copy_u8PinNumber,u8 Copy_u8PortNumber, u8 Copy_u8PinDirection)
{
    if((Copy_u8PinNumber>=0)&&(Copy_u8PinNumber<8))
    {
        switch(Copy_u8PortNumber)
        {
        case MDIO_PORTA:
            if(Copy_u8PinDirection==PIN_OUTPUT)
            {
                SET_BIT(DDRA,Copy_u8PinNumber);
            }
            else if(Copy_u8PinDirection==PIN_INPUT)
            {
                CLR_BIT(DDRA,Copy_u8PinNumber);
            }
            else
            {
                //Invalid Pin Direction
            }
            break;
        case MDIO_PORTB:
            if(Copy_u8PinDirection==PIN_OUTPUT)
            {
                SET_BIT(DDRB,Copy_u8PinNumber);
            }
            else if(Copy_u8PinDirection==PIN_INPUT)
            {
                CLR_BIT(DDRB,Copy_u8PinNumber);
            }
            else
            {
                //Invalid Pin Direction
            }
            break;
```



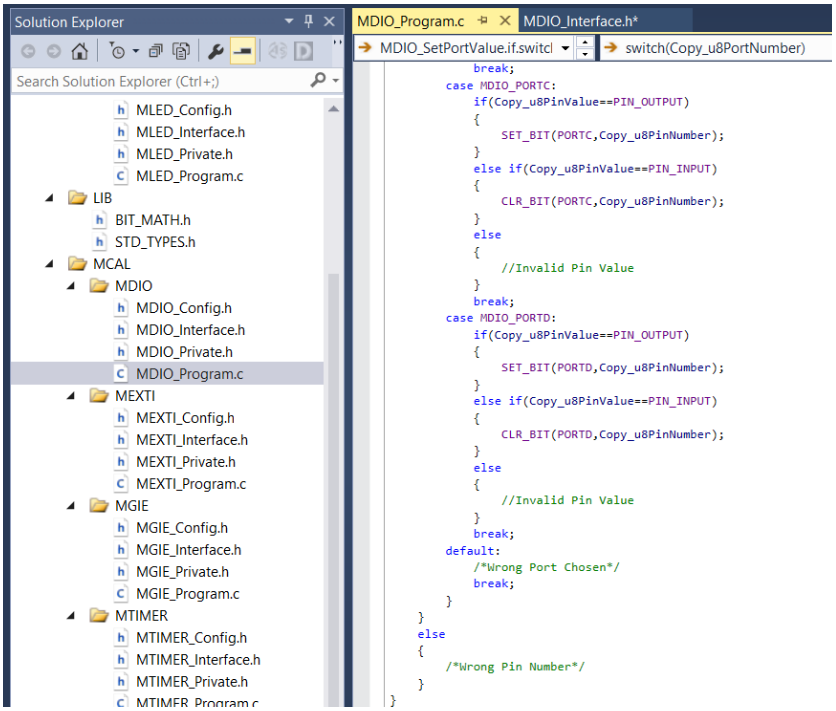
```
            break;
        case MDIO_PORTC:
            if(Copy_u8PinDirection==PIN_OUTPUT)
            {
                SET_BIT(DDRC,Copy_u8PinNumber);
            }
            else if(Copy_u8PinDirection==PIN_INPUT)
            {
                CLR_BIT(DDRC,Copy_u8PinNumber);
            }
            else
            {
                //Invalid Pin Direction
            }
            break;
        case MDIO_PORTD:
            if(Copy_u8PinDirection==PIN_OUTPUT)
            {
                SET_BIT(DDRD,Copy_u8PinNumber);
            }
            else if(Copy_u8PinDirection==PIN_INPUT)
            {
                CLR_BIT(DDRD,Copy_u8PinNumber);
            }
            else
            {
                //Invalid Pin Direction
            }
            break;
        default:
            /*Wrong Port Chosen*/
            break;
        }
    }
    else
    {
        /*Wrong Pin Number*/
    }
}
```



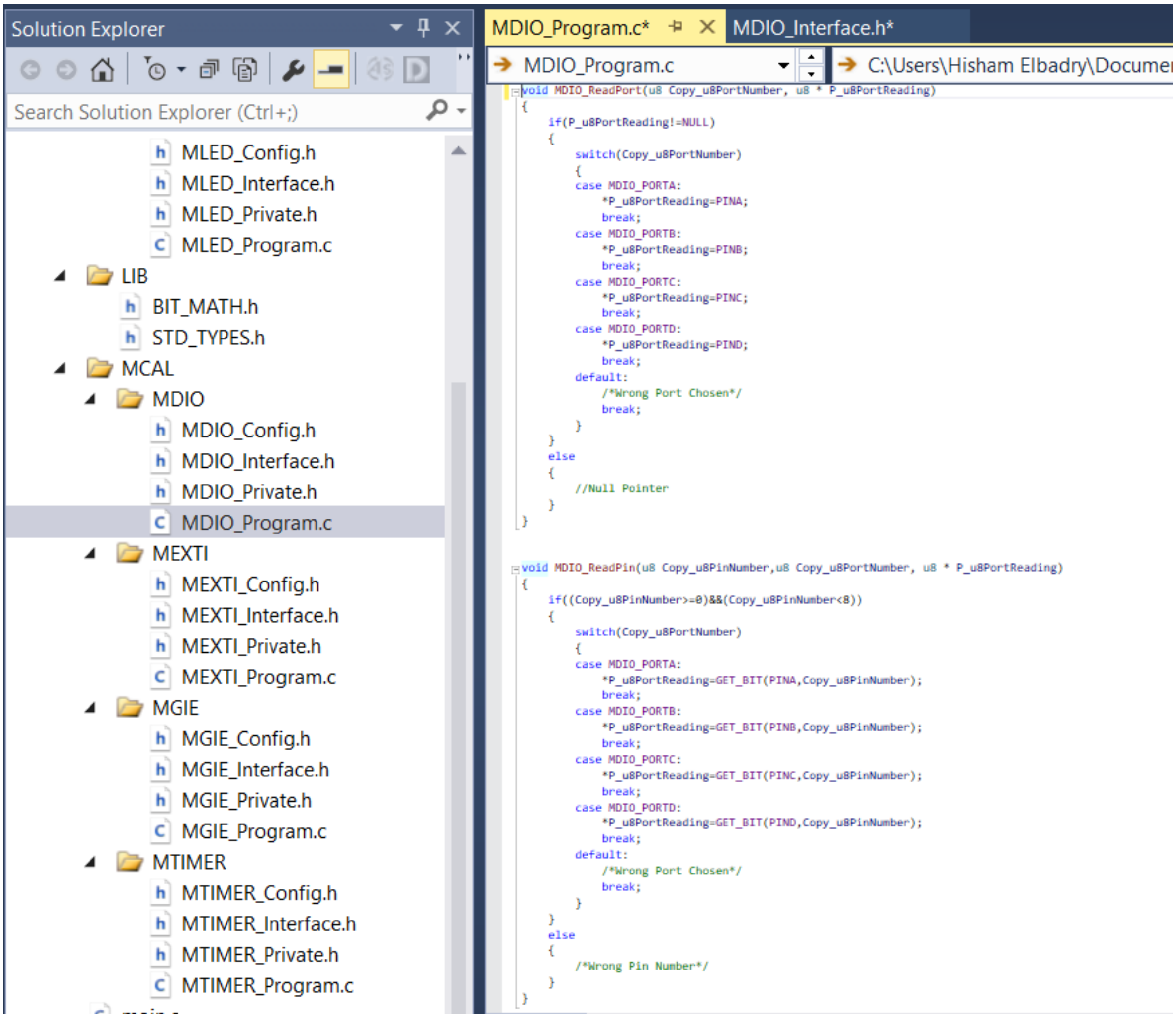
```
void MDIO_SetPortValue(u8 Copy_u8PortNumber, u8 Copy_u8PortValue)
{
    if((Copy_u8PortValue==PORT_HIGH)||(Copy_u8PortValue==PORT_LOW))
    {
        /*Valid Port Direction given */
        switch(Copy_u8PortNumber)
        {
        case MDIO_PORTA:
            PORTA=Copy_u8PortValue;
            break;
        case MDIO_PORTB:
            PORTB=Copy_u8PortValue;
            break;
        case MDIO_PORTC:
            PORTC=Copy_u8PortValue;
            break;
        case MDIO_PORTD:
            PORTD=Copy_u8PortValue;
            break;
        default:
            /*Wrong Port Number Passed*/
            break;
        }
    }
    else
    {
        /*Invalid Port Value given */
    }
}
```



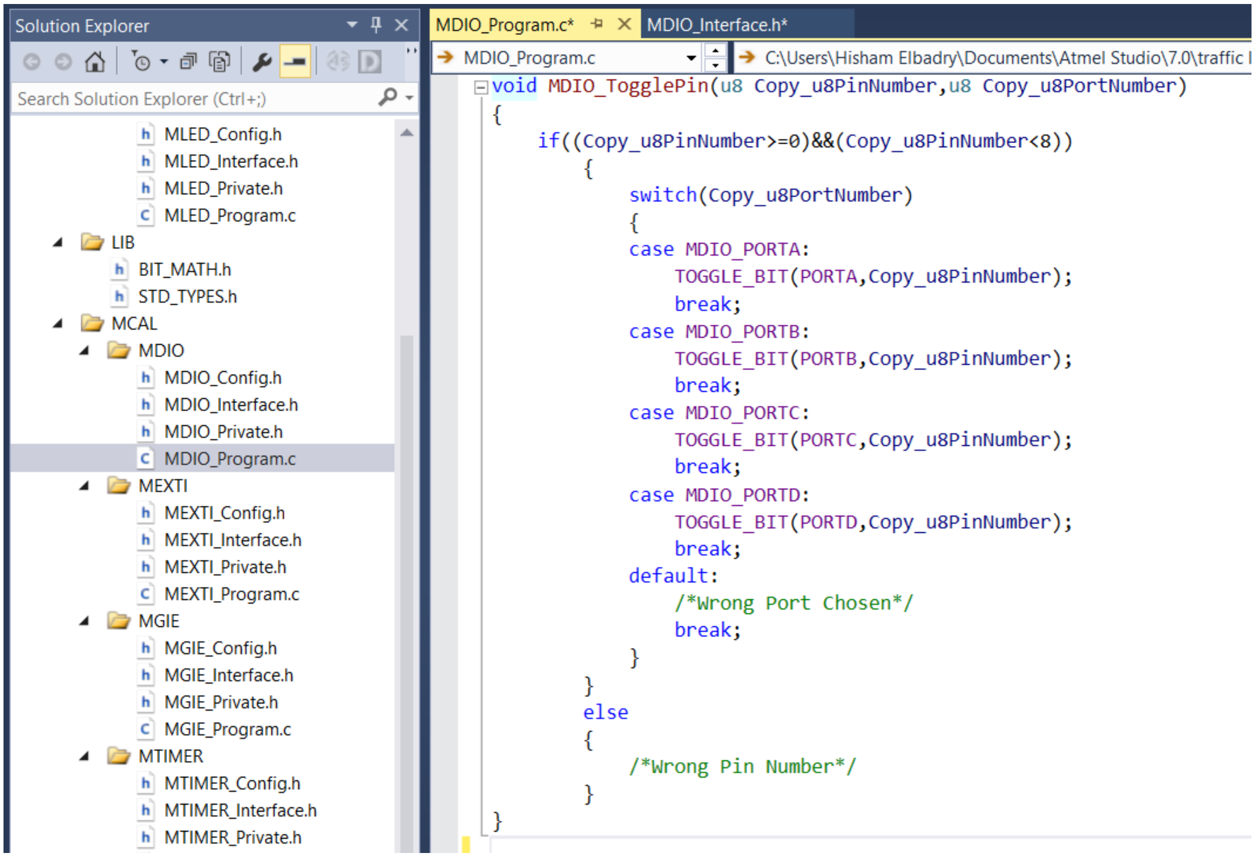
```
void MDIO_SetPinValue(u8 Copy_u8PinNumber,u8 Copy_u8PortNumber, u8 Copy_u8PinValue)
{
    if((Copy_u8PinNumber>=0)&&(Copy_u8PinNumber<8))
    {
        switch(Copy_u8PortNumber)
        {
        case MDIO_PORTA:
            if(Copy_u8PinValue==PIN_HIGH)
            {
                SET_BIT(PORTA,Copy_u8PinNumber);
            }
            else if(Copy_u8PinValue==PIN_LOW)
            {
                CLR_BIT(PORTA,Copy_u8PinNumber);
            }
            else
            {
                //Invalid Pin Value
            }
            break;
        case MDIO_PORTB:
            if(Copy_u8PinValue==PIN_HIGH)
            {
                SET_BIT(PORTB,Copy_u8PinNumber);
            }
            else if(Copy_u8PinValue==PIN_INPUT)
            {
                CLR_BIT(PORTB,Copy_u8PinNumber);
            }
            else
            {
                //Invalid Pin Value
            }
            break;
```



```
                break;
            case MDIO_PORTC:
                if(Copy_u8PinValue==PIN_OUTPUT)
                {
                    SET_BIT(PORTC,Copy_u8PinNumber);
                }
                else if(Copy_u8PinValue==PIN_INPUT)
                {
                    CLR_BIT(PORTC,Copy_u8PinNumber);
                }
                else
                {
                    //Invalid Pin Value
                }
                break;
            case MDIO_PORTD:
                if(Copy_u8PinValue==PIN_OUTPUT)
                {
                    SET_BIT(PORTD,Copy_u8PinNumber);
                }
                else if(Copy_u8PinValue==PIN_INPUT)
                {
                    CLR_BIT(PORTD,Copy_u8PinNumber);
                }
                else
                {
                    //Invalid Pin Value
                }
                break;
            default:
                /*Wrong Port Chosen*/
                break;
            }
        }
        else
        {
            /*Wrong Pin Number*/
        }
}
```

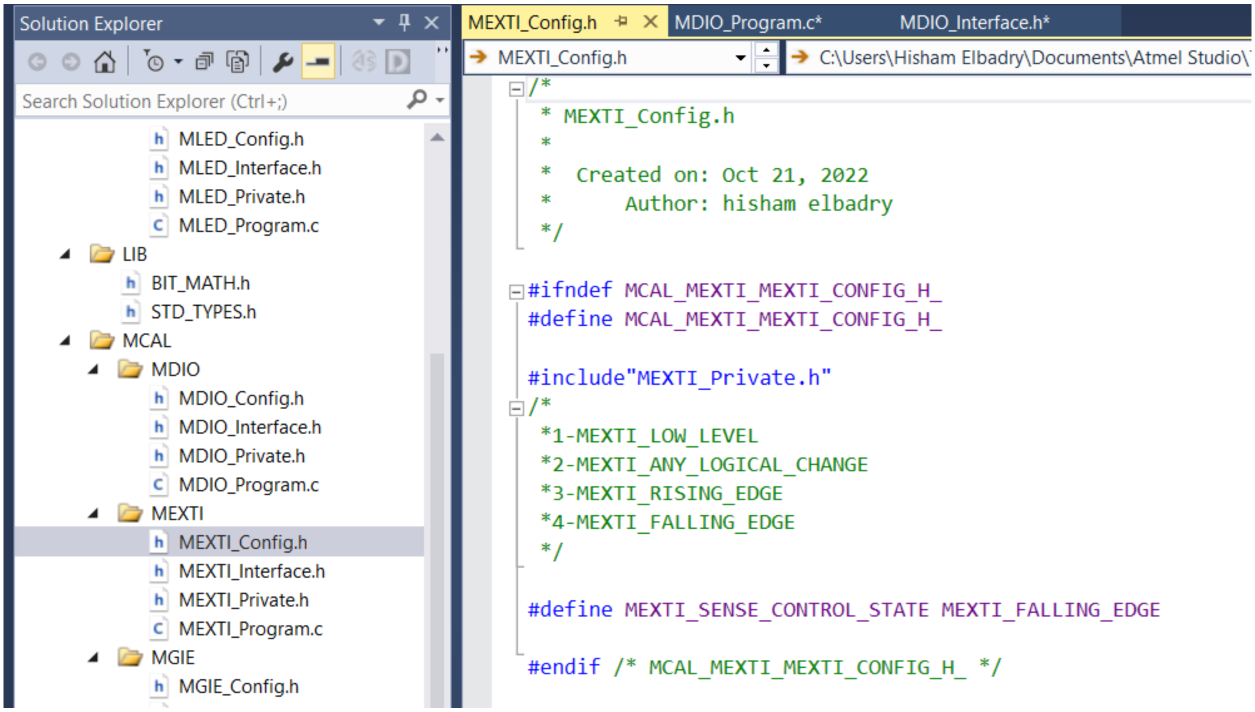


```
void MDIO_ReadPort(u8 Copy_u8PortNumber, u8 * P_u8PortReading)
{
    if(P_u8PortReading!=NULL)
    {
        switch(Copy_u8PortNumber)
        {
        case MDIO_PORTA:
            *P_u8PortReading=PINA;
            break;
        case MDIO_PORTB:
            *P_u8PortReading=PINB;
            break;
        case MDIO_PORTC:
            *P_u8PortReading=PINC;
            break;
        case MDIO_PORTD:
            *P_u8PortReading=PIND;
            break;
        default:
            /*Wrong Port Chosen*/
            break;
        }
    }
    else
    {
        //Null Pointer
    }
}


void MDIO_ReadPin(u8 Copy_u8PinNumber,u8 Copy_u8PortNumber, u8 * P_u8PortReading)
{
    if((Copy_u8PinNumber>=0)&&(Copy_u8PinNumber<8))
    {
        switch(Copy_u8PortNumber)
        {
        case MDIO_PORTA:
            *P_u8PortReading=GET_BIT(PINA,Copy_u8PinNumber);
            break;
        case MDIO_PORTB:
            *P_u8PortReading=GET_BIT(PINB,Copy_u8PinNumber);
            break;
        case MDIO_PORTC:
            *P_u8PortReading=GET_BIT(PINC,Copy_u8PinNumber);
            break;
        case MDIO_PORTD:
            *P_u8PortReading=GET_BIT(PIND,Copy_u8PinNumber);
            break;
        default:
            /*Wrong Port Chosen*/
            break;
        }
    }
    else
    {
        /*Wrong Pin Number*/
    }
}
```

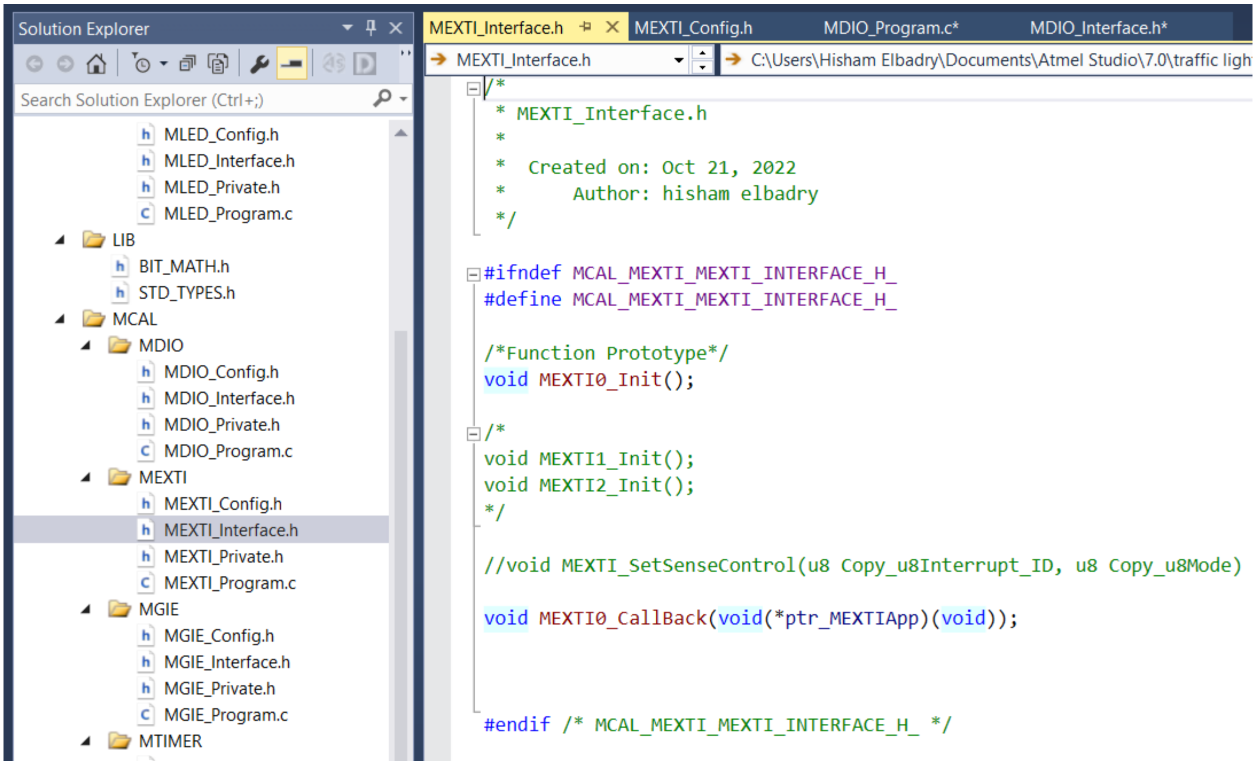


```c
void MDIO_TogglePin(u8 Copy_u8PinNumber,u8 Copy_u8PortNumber)
{
    if((Copy_u8PinNumber>=0)&&(Copy_u8PinNumber<8))
        {
            switch(Copy_u8PortNumber)
            {
            case MDIO_PORTA:
                TOGGLE_BIT(PORTA,Copy_u8PinNumber);
                break;
            case MDIO_PORTB:
                TOGGLE_BIT(PORTB,Copy_u8PinNumber);
                break;
            case MDIO_PORTC:
                TOGGLE_BIT(PORTC,Copy_u8PinNumber);
                break;
            case MDIO_PORTD:
                TOGGLE_BIT(PORTD,Copy_u8PinNumber);
                break;
            default:
                /*Wrong Port Chosen*/
                break;
            }
        }
        else
        {
            /*Wrong Pin Number*/
        }
}
```

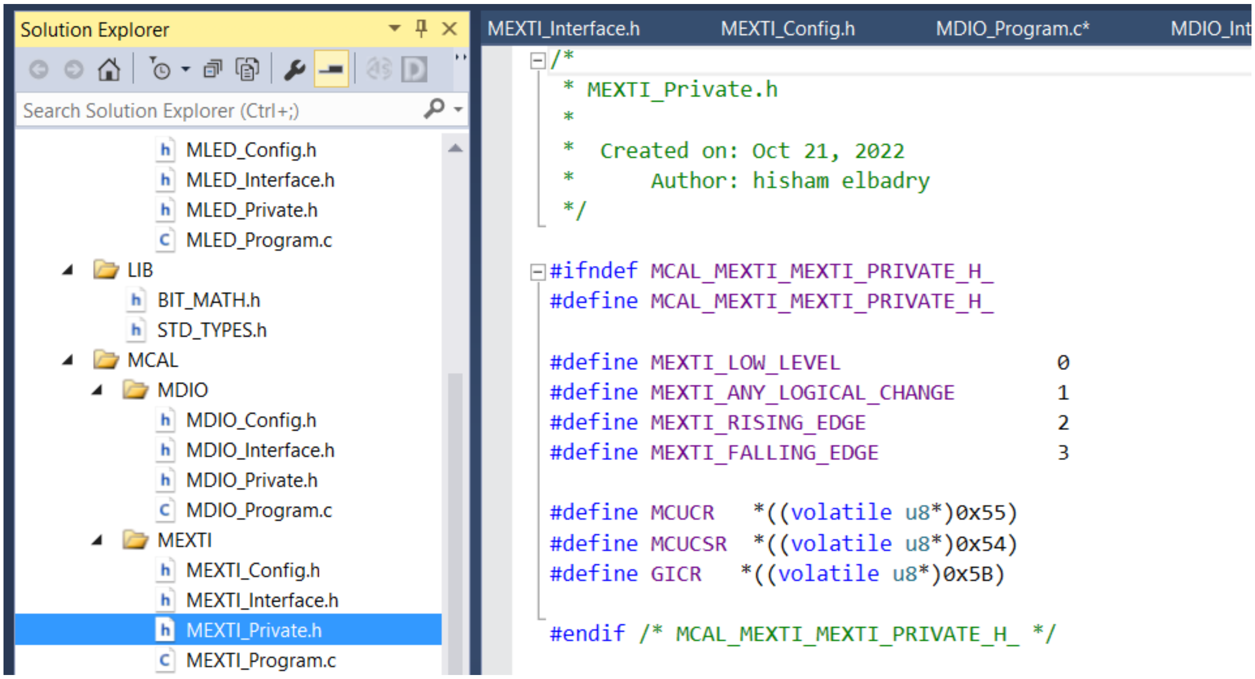
External interrupt files



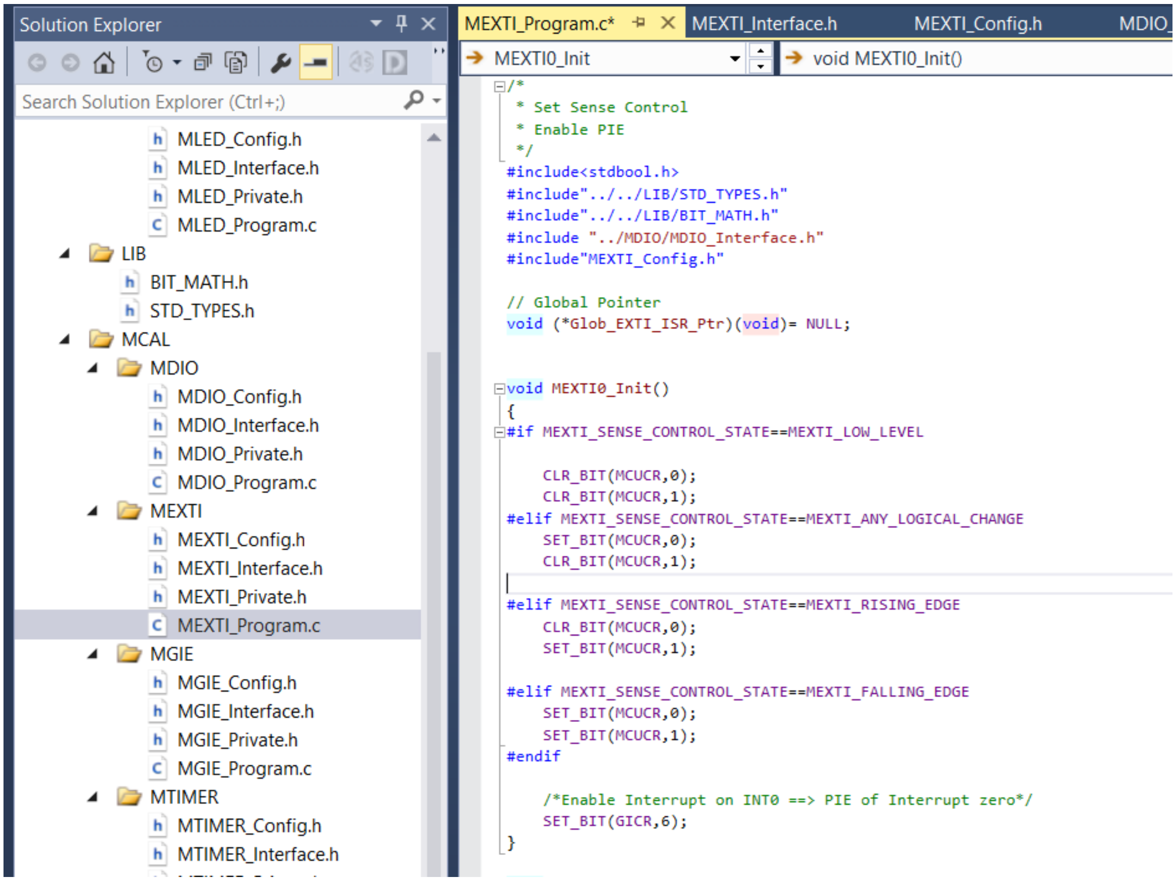
```c
/*
 * MEXTI_Config.h
 *
 *  Created on: Oct 21, 2022
 *      Author: hisham elbadry
 */

#ifndef MCAL_MEXTI_MEXTI_CONFIG_H_
#define MCAL_MEXTI_MEXTI_CONFIG_H_

#include"MEXTI_Private.h"
/*
 *1-MEXTI_LOW_LEVEL
 *2-MEXTI_ANY_LOGICAL_CHANGE
 *3-MEXTI_RISING_EDGE
 *4-MEXTI_FALLING_EDGE
 */

#define MEXTI_SENSE_CONTROL_STATE MEXTI_FALLING_EDGE

#endif /* MCAL_MEXTI_MEXTI_CONFIG_H_ */
```



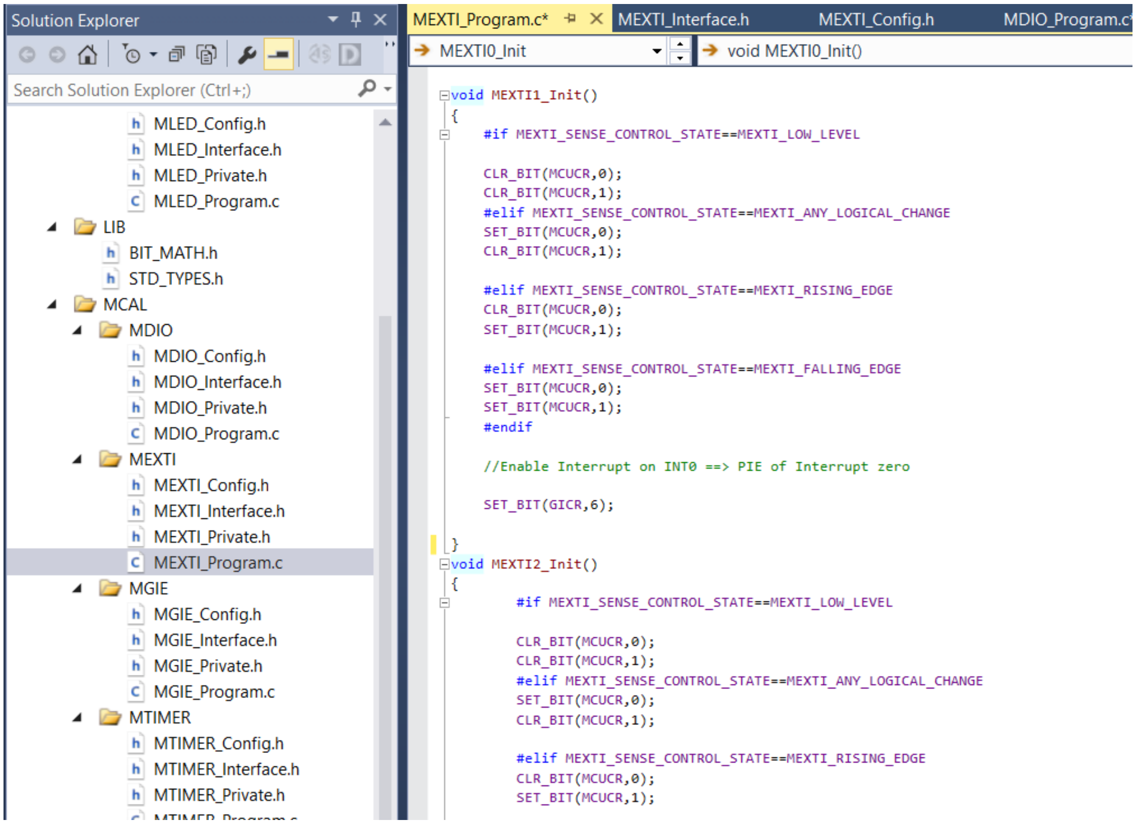
```c
/*
 * MEXTI_Interface.h
 *
 *  Created on: Oct 21, 2022
 *      Author: hisham elbadry
 */

#ifndef MCAL_MEXTI_MEXTI_INTERFACE_H_
#define MCAL_MEXTI_MEXTI_INTERFACE_H_

/*Function Prototype*/
void MEXTI0_Init();

/*
void MEXTI1_Init();
void MEXTI2_Init();
*/

//void MEXTI_SetSenseControl(u8 Copy_u8Interrupt_ID, u8 Copy_u8Mode)

void MEXTI0_CallBack(void(*ptr_MEXTIApp)(void));



#endif /* MCAL_MEXTI_MEXTI_INTERFACE_H_ */
```



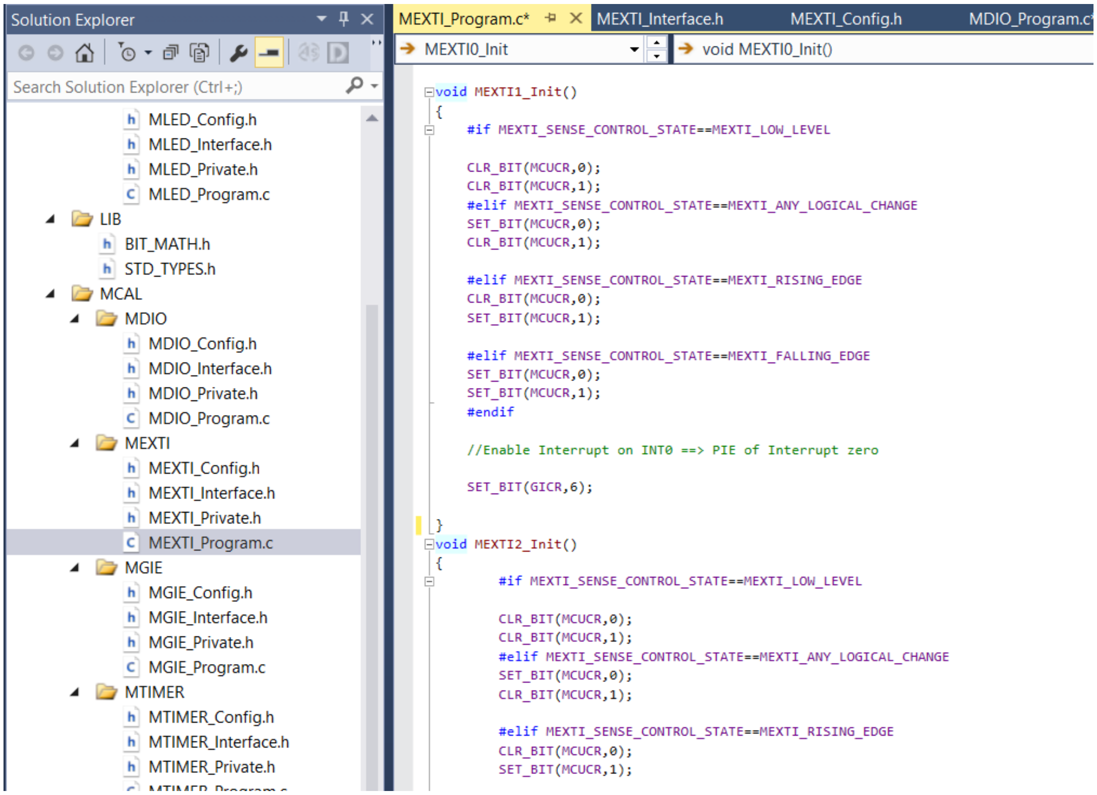
```c
/*
 * MEXTI_Private.h
 *
 *  Created on: Oct 21, 2022
 *      Author: hisham elbadry
 */

#ifndef MCAL_MEXTI_MEXTI_PRIVATE_H_
#define MCAL_MEXTI_MEXTI_PRIVATE_H_

#define MEXTI_LOW_LEVEL                 0
#define MEXTI_ANY_LOGICAL_CHANGE        1
#define MEXTI_RISING_EDGE               2
#define MEXTI_FALLING_EDGE              3

#define MCUCR   *((volatile u8*)0x55)
#define MCUCSR  *((volatile u8*)0x54)
#define GICR   *((volatile u8*)0x5B)

#endif /* MCAL_MEXTI_MEXTI_PRIVATE_H_ */
```



```c
/*
 * Set Sense Control
 * Enable PIE
 */
#include<stdbool.h>
#include"../../LIB/STD_TYPES.h"
#include"../../LIB/BIT_MATH.h"
#include "../MDIO/MDIO_Interface.h"
#include"MEXTI_Config.h"

// Global Pointer
void (*Glob_EXTI_ISR_Ptr)(void)= NULL;


void MEXTI0_Init()
{
#if MEXTI_SENSE_CONTROL_STATE==MEXTI_LOW_LEVEL

    CLR_BIT(MCUCR,0);
    CLR_BIT(MCUCR,1);
#elif MEXTI_SENSE_CONTROL_STATE==MEXTI_ANY_LOGICAL_CHANGE
    SET_BIT(MCUCR,0);
    CLR_BIT(MCUCR,1);

#elif MEXTI_SENSE_CONTROL_STATE==MEXTI_RISING_EDGE
    CLR_BIT(MCUCR,0);
    SET_BIT(MCUCR,1);

#elif MEXTI_SENSE_CONTROL_STATE==MEXTI_FALLING_EDGE
    SET_BIT(MCUCR,0);
    SET_BIT(MCUCR,1);
#endif

    /*Enable Interrupt on INT0 ==> PIE of Interrupt zero*/
    SET_BIT(GICR,6);
}
```

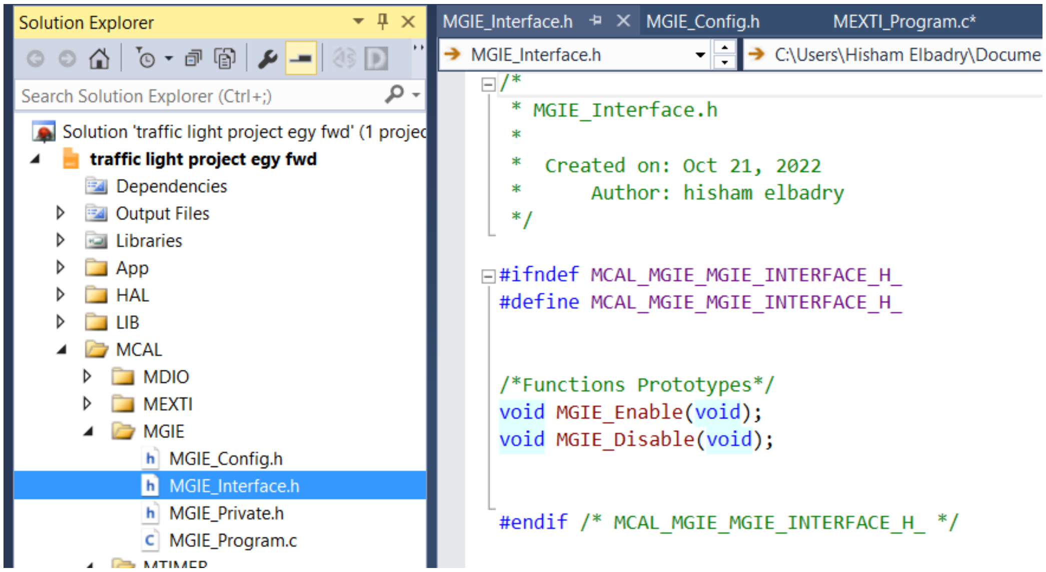


```
void MEXTI1_Init()
{
    #if MEXTI_SENSE_CONTROL_STATE==MEXTI_LOW_LEVEL

    CLR_BIT(MCUCR,0);
    CLR_BIT(MCUCR,1);
    #elif MEXTI_SENSE_CONTROL_STATE==MEXTI_ANY_LOGICAL_CHANGE
    SET_BIT(MCUCR,0);
    CLR_BIT(MCUCR,1);

    #elif MEXTI_SENSE_CONTROL_STATE==MEXTI_RISING_EDGE
    CLR_BIT(MCUCR,0);
    SET_BIT(MCUCR,1);

    #elif MEXTI_SENSE_CONTROL_STATE==MEXTI_FALLING_EDGE
    SET_BIT(MCUCR,0);
    SET_BIT(MCUCR,1);
    #endif

    //Enable Interrupt on INT0 ==> PIE of Interrupt zero

    SET_BIT(GICR,6);

}
void MEXTI2_Init()
{
        #if MEXTI_SENSE_CONTROL_STATE==MEXTI_LOW_LEVEL

        CLR_BIT(MCUCR,0);
        CLR_BIT(MCUCR,1);
        #elif MEXTI_SENSE_CONTROL_STATE==MEXTI_ANY_LOGICAL_CHANGE
        SET_BIT(MCUCR,0);
        CLR_BIT(MCUCR,1);

        #elif MEXTI_SENSE_CONTROL_STATE==MEXTI_RISING_EDGE
        CLR_BIT(MCUCR,0);
        SET_BIT(MCUCR,1);
```

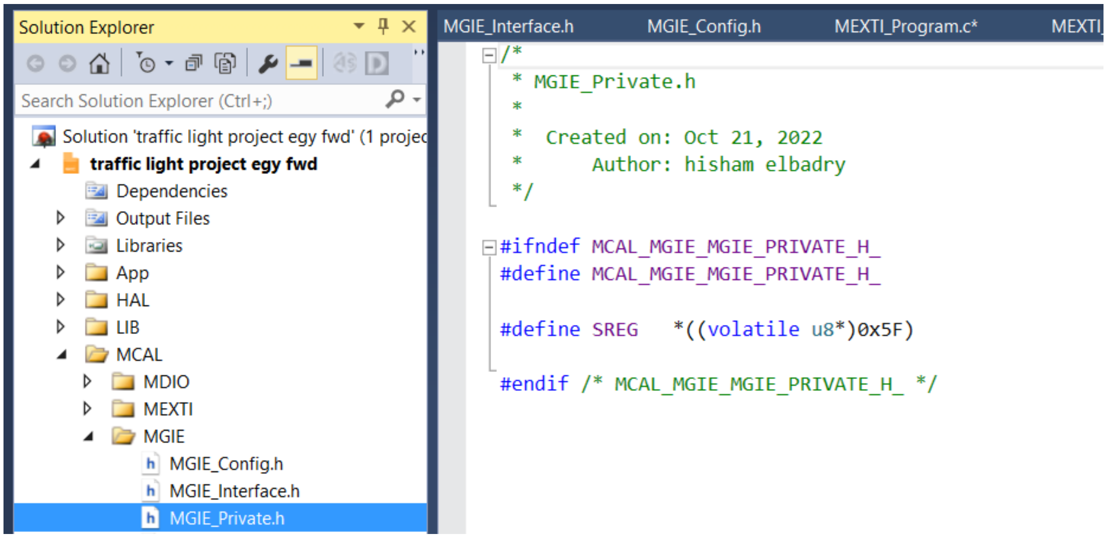


```
void MEXTI1_Init()
{
    #if MEXTI_SENSE_CONTROL_STATE==MEXTI_LOW_LEVEL

    CLR_BIT(MCUCR,0);
    CLR_BIT(MCUCR,1);
    #elif MEXTI_SENSE_CONTROL_STATE==MEXTI_ANY_LOGICAL_CHANGE
    SET_BIT(MCUCR,0);
    CLR_BIT(MCUCR,1);

    #elif MEXTI_SENSE_CONTROL_STATE==MEXTI_RISING_EDGE
    CLR_BIT(MCUCR,0);
    SET_BIT(MCUCR,1);

    #elif MEXTI_SENSE_CONTROL_STATE==MEXTI_FALLING_EDGE
    SET_BIT(MCUCR,0);
    SET_BIT(MCUCR,1);
    #endif

    //Enable Interrupt on INT0 ==> PIE of Interrupt zero

    SET_BIT(GICR,6);

}
void MEXTI2_Init()
{
        #if MEXTI_SENSE_CONTROL_STATE==MEXTI_LOW_LEVEL

        CLR_BIT(MCUCR,0);
        CLR_BIT(MCUCR,1);
        #elif MEXTI_SENSE_CONTROL_STATE==MEXTI_ANY_LOGICAL_CHANGE
        SET_BIT(MCUCR,0);
        CLR_BIT(MCUCR,1);

        #elif MEXTI_SENSE_CONTROL_STATE==MEXTI_RISING_EDGE
        CLR_BIT(MCUCR,0);
        SET_BIT(MCUCR,1);
```

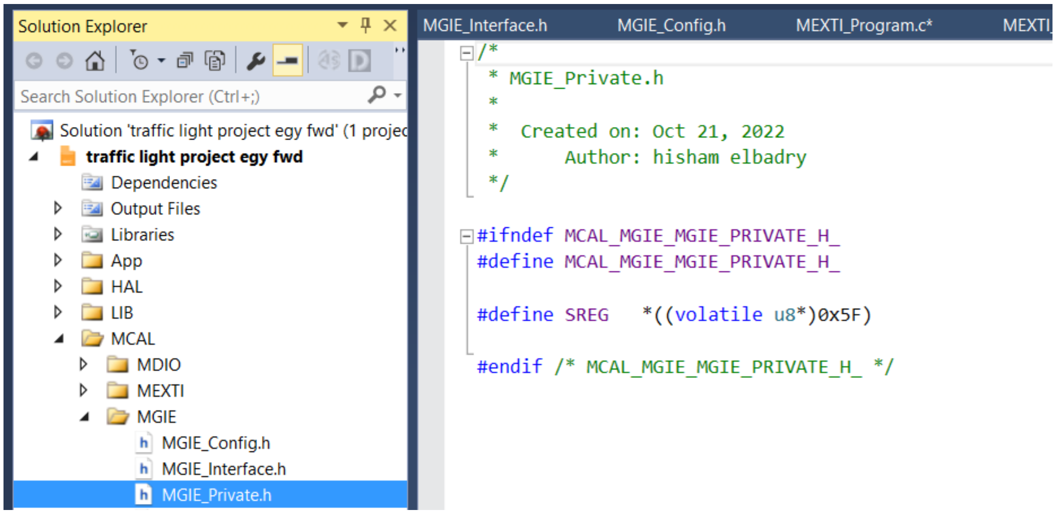
Internal interrupt files



```
/*
 * MGIE_Interface.h
 *
 *  Created on: Oct 21, 2022
 *      Author: hisham elbadry
 */

#ifndef MCAL_MGIE_MGIE_INTERFACE_H_
#define MCAL_MGIE_MGIE_INTERFACE_H_


/*Functions Prototypes*/
void MGIE_Enable(void);
void MGIE_Disable(void);


#endif /* MCAL_MGIE_MGIE_INTERFACE_H_ */
```

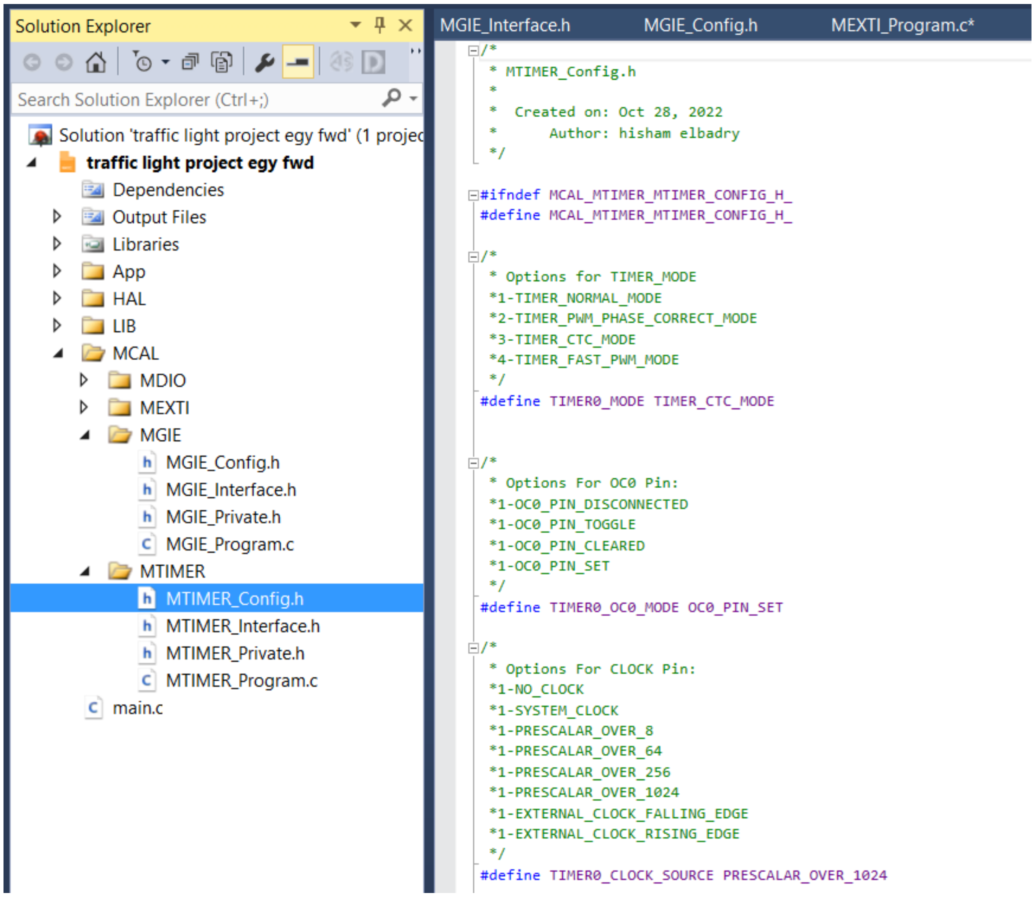


```
/*
 * MGIE_Private.h
 *
 *  Created on: Oct 21, 2022
 *      Author: hisham elbadry
 */

#ifndef MCAL_MGIE_MGIE_PRIVATE_H_
#define MCAL_MGIE_MGIE_PRIVATE_H_

#define SREG   *((volatile u8*)0x5F)

#endif /* MCAL_MGIE_MGIE_PRIVATE_H_ */
```

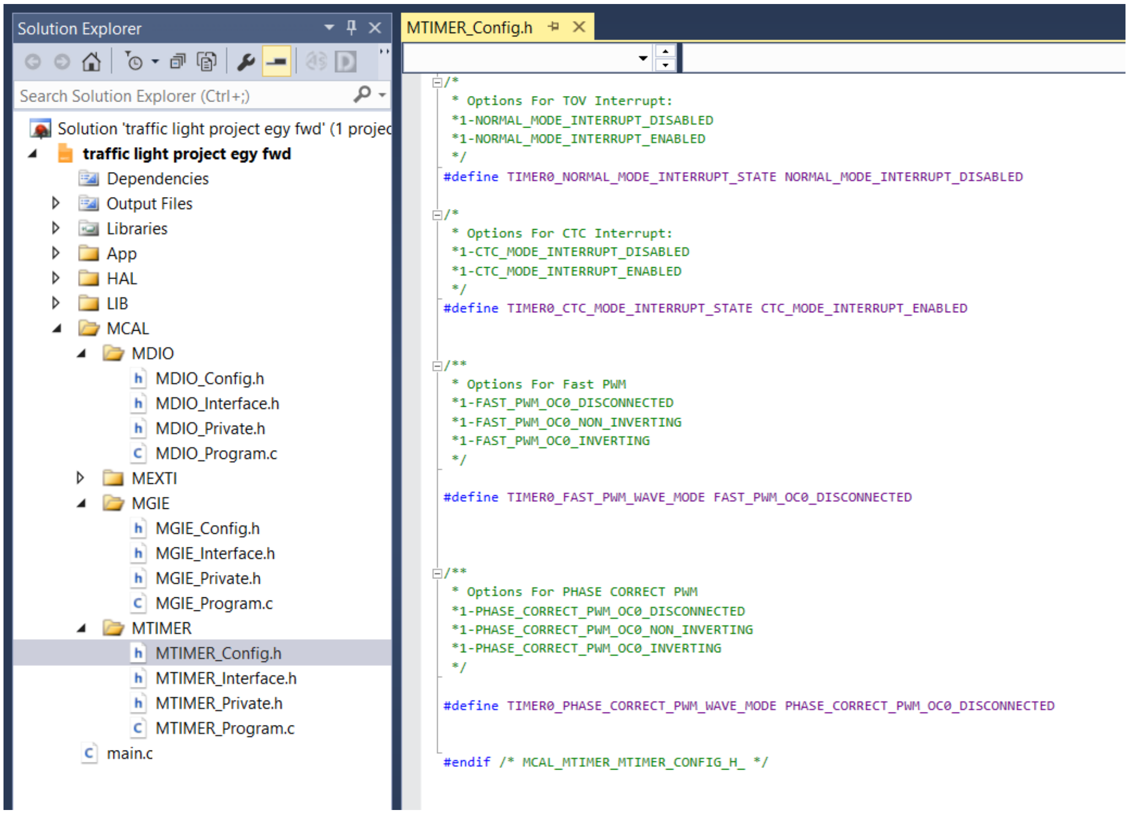


```
/*
 * MGIE_Private.h
 *
 *  Created on: Oct 21, 2022
 *      Author: hisham elbadry
 */

#ifndef MCAL_MGIE_MGIE_PRIVATE_H_
#define MCAL_MGIE_MGIE_PRIVATE_H_

#define SREG   *((volatile u8*)0x5F)

#endif /* MCAL_MGIE_MGIE_PRIVATE_H_ */
```

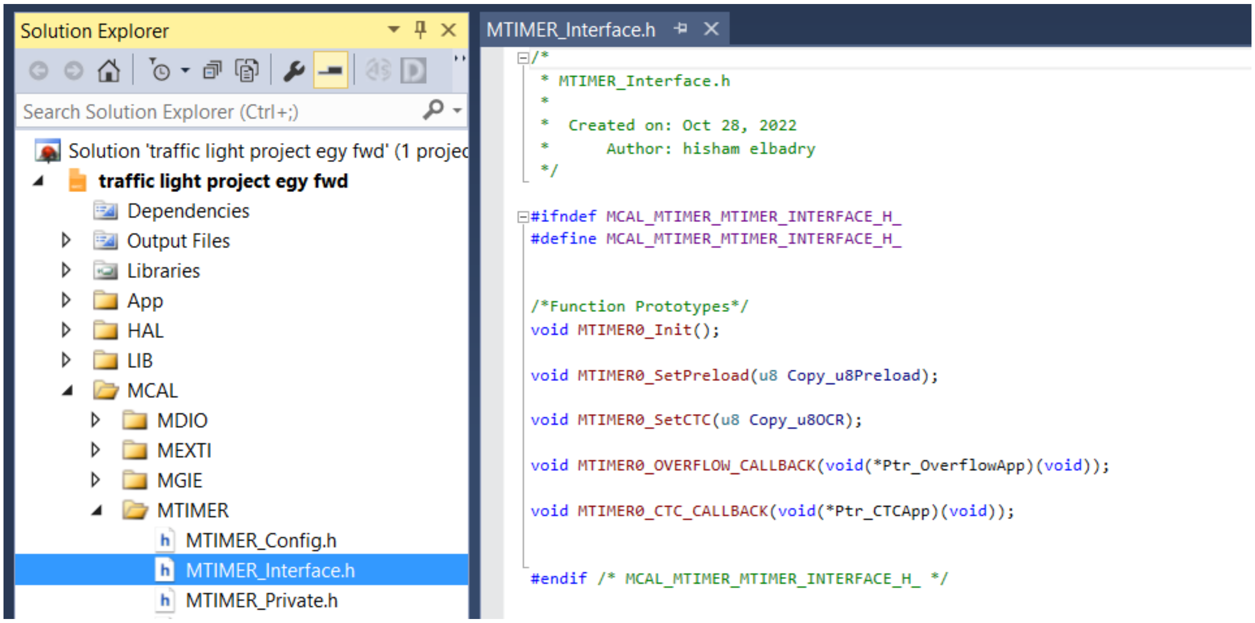
Timer files



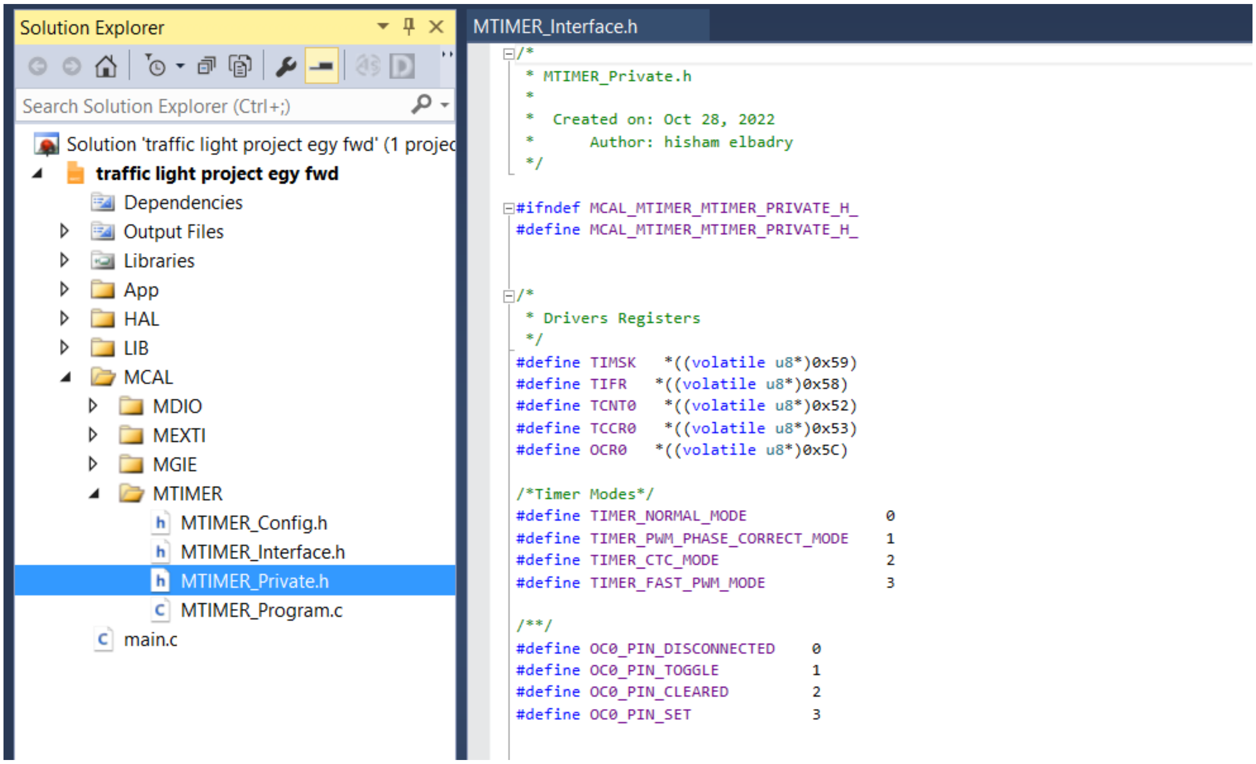
```c
/*
 * MTIMER_Config.h
 *
 *  Created on: Oct 28, 2022
 *      Author: hisham elbadry
 */

#ifndef MCAL_MTIMER_MTIMER_CONFIG_H_
#define MCAL_MTIMER_MTIMER_CONFIG_H_

/*
 * Options for TIMER_MODE
 *1-TIMER_NORMAL_MODE
 *2-TIMER_PWM_PHASE_CORRECT_MODE
 *3-TIMER_CTC_MODE
 *4-TIMER_FAST_PWM_MODE
 */
#define TIMER0_MODE TIMER_CTC_MODE


/*
 * Options For OC0 Pin:
 *1-OC0_PIN_DISCONNECTED
 *1-OC0_PIN_TOGGLE
 *1-OC0_PIN_CLEARED
 *1-OC0_PIN_SET
 */
#define TIMER0_OC0_MODE OC0_PIN_SET

/*
 * Options For CLOCK Pin:
 *1-NO_CLOCK
 *1-SYSTEM_CLOCK
 *1-PRESCALAR_OVER_8
 *1-PRESCALAR_OVER_64
 *1-PRESCALAR_OVER_256
 *1-PRESCALAR_OVER_1024
 *1-EXTERNAL_CLOCK_FALLING_EDGE
 *1-EXTERNAL_CLOCK_RISING_EDGE
 */
#define TIMER0_CLOCK_SOURCE PRESCALAR_OVER_1024
```



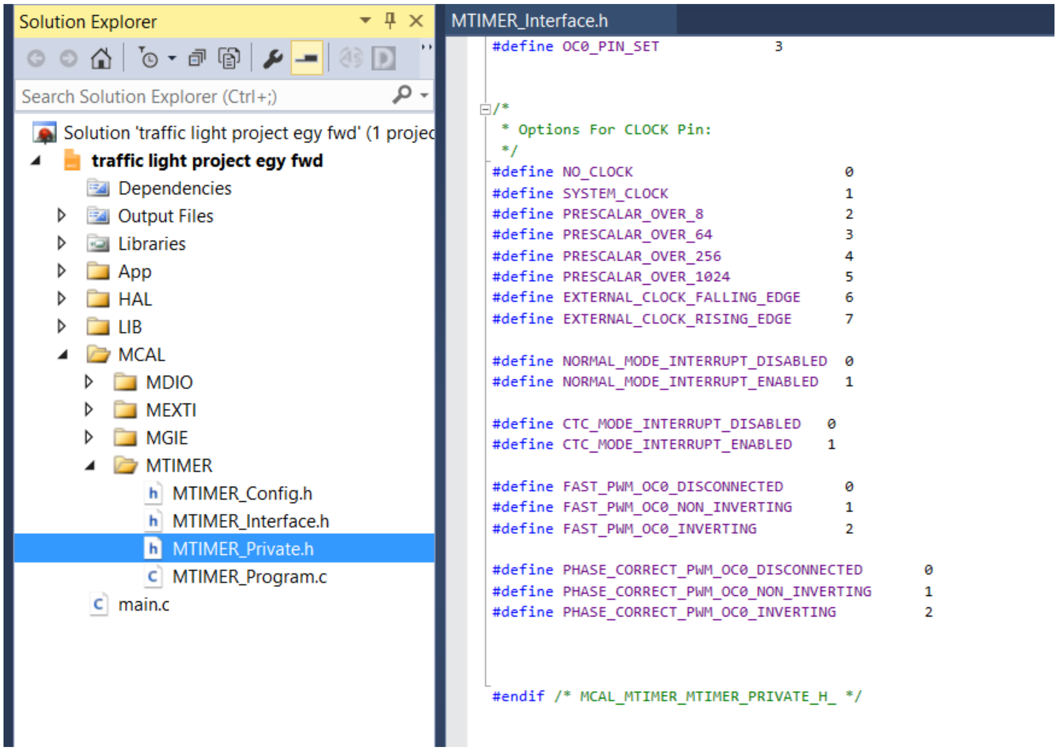
```c
/*
 * Options For TOV Interrupt:
 *1-NORMAL_MODE_INTERRUPT_DISABLED
 *1-NORMAL_MODE_INTERRUPT_ENABLED
 */
#define TIMER0_NORMAL_MODE_INTERRUPT_STATE NORMAL_MODE_INTERRUPT_DISABLED

/*
 * Options For CTC Interrupt:
 *1-CTC_MODE_INTERRUPT_DISABLED
 *1-CTC_MODE_INTERRUPT_ENABLED
 */
#define TIMER0_CTC_MODE_INTERRUPT_STATE CTC_MODE_INTERRUPT_ENABLED


/**
 * Options For Fast PWM
 *1-FAST_PWM_OC0_DISCONNECTED
 *1-FAST_PWM_OC0_NON_INVERTING
 *1-FAST_PWM_OC0_INVERTING
 */

#define TIMER0_FAST_PWM_WAVE_MODE FAST_PWM_OC0_DISCONNECTED



/**
 * Options For PHASE CORRECT PWM
 *1-PHASE_CORRECT_PWM_OC0_DISCONNECTED
 *1-PHASE_CORRECT_PWM_OC0_NON_INVERTING
 *1-PHASE_CORRECT_PWM_OC0_INVERTING
 */

#define TIMER0_PHASE_CORRECT_PWM_WAVE_MODE PHASE_CORRECT_PWM_OC0_DISCONNECTED


#endif /* MCAL_MTIMER_MTIMER_CONFIG_H_ */
```



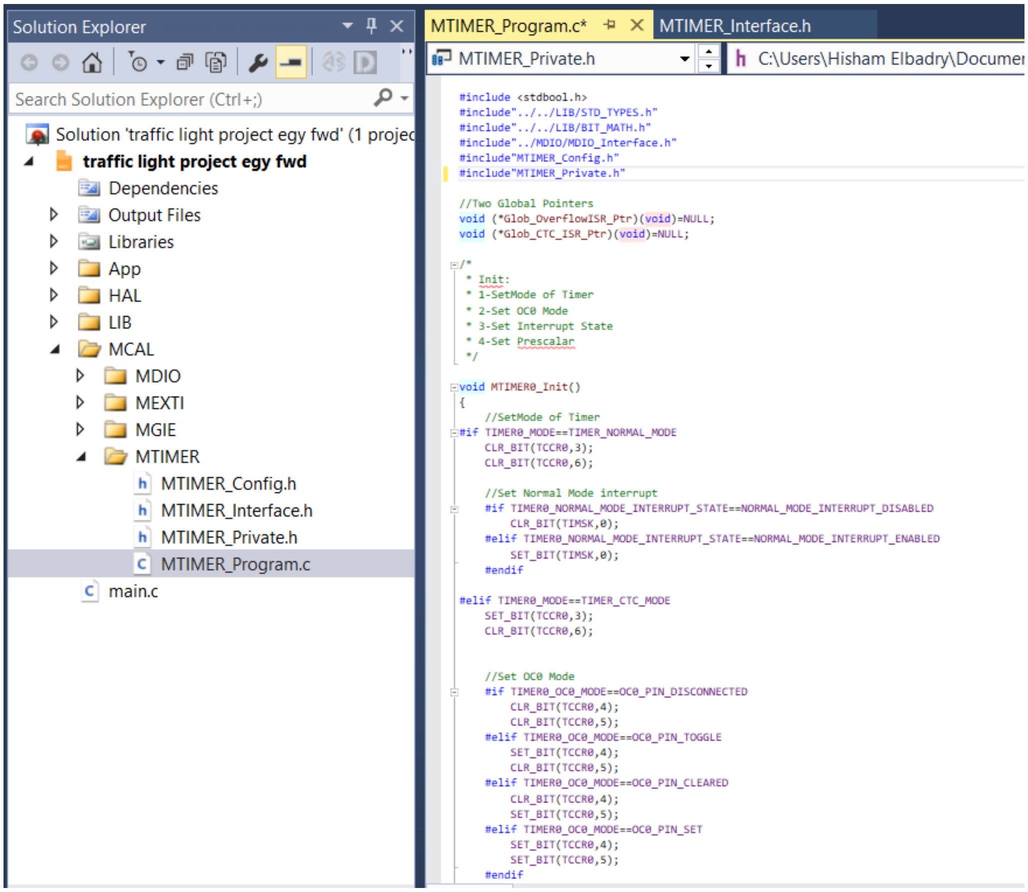
```c
/*
 * MTIMER_Interface.h
 *
 *  Created on: Oct 28, 2022
 *      Author: hisham elbadry
 */

#ifndef MCAL_MTIMER_MTIMER_INTERFACE_H_
#define MCAL_MTIMER_MTIMER_INTERFACE_H_


/*Function Prototypes*/
void MTIMER0_Init();

void MTIMER0_SetPreload(u8 Copy_u8Preload);

void MTIMER0_SetCTC(u8 Copy_u8OCR);

void MTIMER0_OVERFLOW_CALLBACK(void(*Ptr_OverflowApp)(void));

void MTIMER0_CTC_CALLBACK(void(*Ptr_CTCApp)(void));


#endif /* MCAL_MTIMER_MTIMER_INTERFACE_H_ */
```



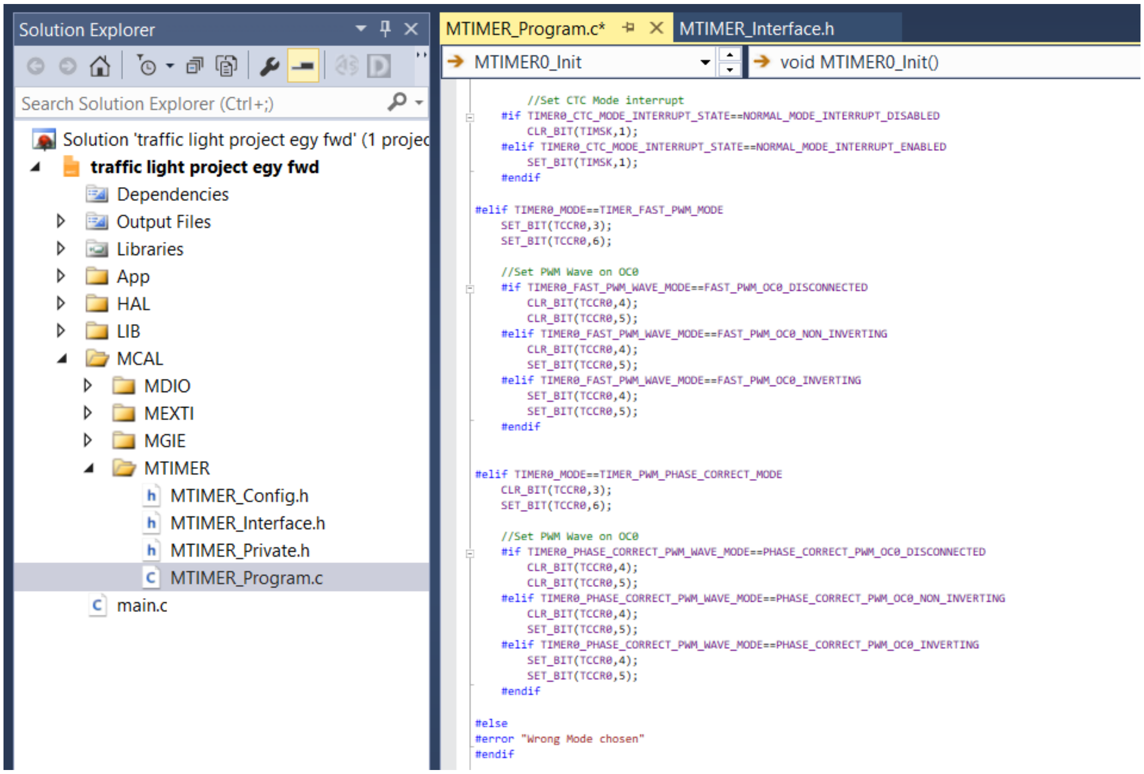
```c
/*
 * MTIMER_Private.h
 *
 *  Created on: Oct 28, 2022
 *      Author: hisham elbadry
 */

#ifndef MCAL_MTIMER_MTIMER_PRIVATE_H_
#define MCAL_MTIMER_MTIMER_PRIVATE_H_


/*
 * Drivers Registers
 */
#define TIMSK   *((volatile u8*)0x59)
#define TIFR   *((volatile u8*)0x58)
#define TCNT0   *((volatile u8*)0x52)
#define TCCR0   *((volatile u8*)0x53)
#define OCR0   *((volatile u8*)0x5C)

/*Timer Modes*/
#define TIMER_NORMAL_MODE               0
#define TIMER_PWM_PHASE_CORRECT_MODE    1
#define TIMER_CTC_MODE                  2
#define TIMER_FAST_PWM_MODE             3

/**/
#define OC0_PIN_DISCONNECTED    0
#define OC0_PIN_TOGGLE          1
#define OC0_PIN_CLEARED         2
#define OC0_PIN_SET             3
```



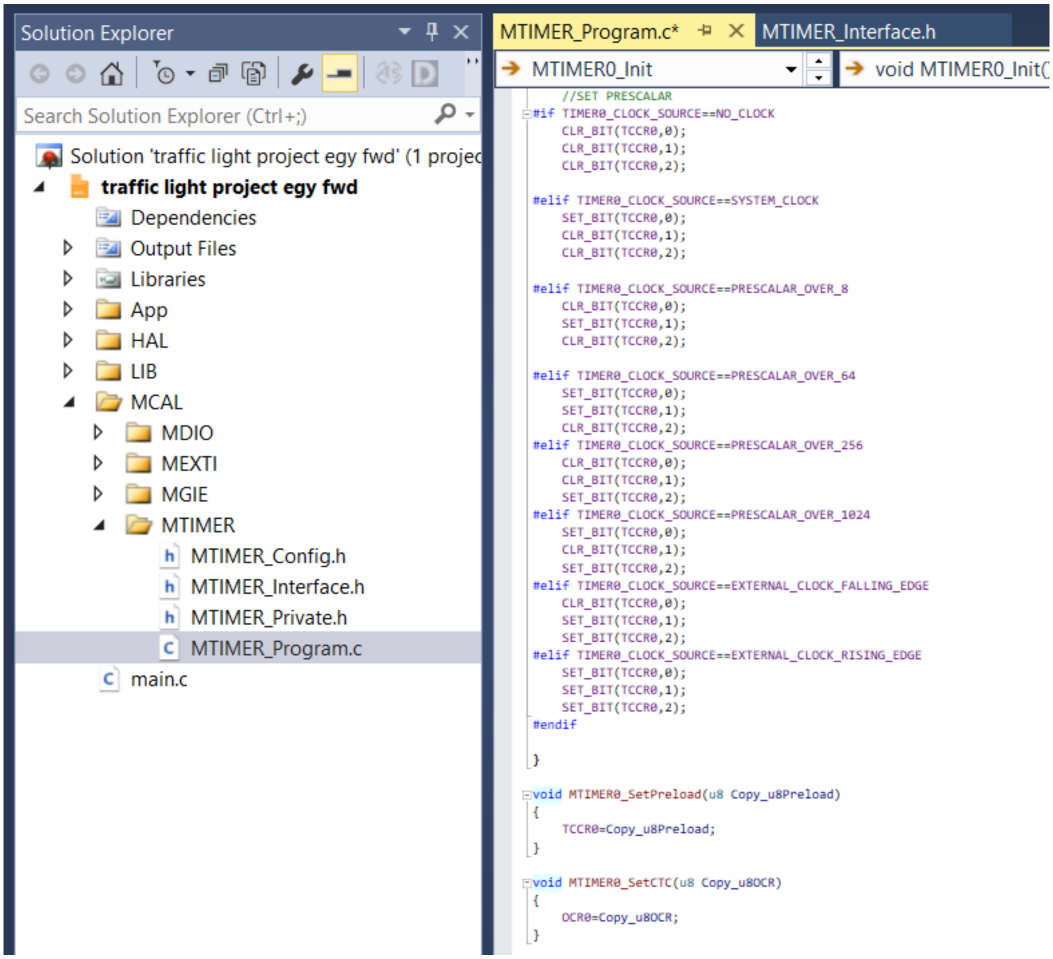
```c
#define OC0_PIN_SET             3

/*
 * Options For CLOCK Pin:
 */
#define NO_CLOCK                        0
#define SYSTEM_CLOCK                    1
#define PRESCALAR_OVER_8                2
#define PRESCALAR_OVER_64               3
#define PRESCALAR_OVER_256              4
#define PRESCALAR_OVER_1024             5
#define EXTERNAL_CLOCK_FALLING_EDGE     6
#define EXTERNAL_CLOCK_RISING_EDGE      7

#define NORMAL_MODE_INTERRUPT_DISABLED  0
#define NORMAL_MODE_INTERRUPT_ENABLED   1

#define CTC_MODE_INTERRUPT_DISABLED   0
#define CTC_MODE_INTERRUPT_ENABLED    1

#define FAST_PWM_OC0_DISCONNECTED       0
#define FAST_PWM_OC0_NON_INVERTING      1
#define FAST_PWM_OC0_INVERTING          2

#define PHASE_CORRECT_PWM_OC0_DISCONNECTED       0
#define PHASE_CORRECT_PWM_OC0_NON_INVERTING      1
#define PHASE_CORRECT_PWM_OC0_INVERTING          2


#endif /* MCAL_MTIMER_MTIMER_PRIVATE_H_ */
```



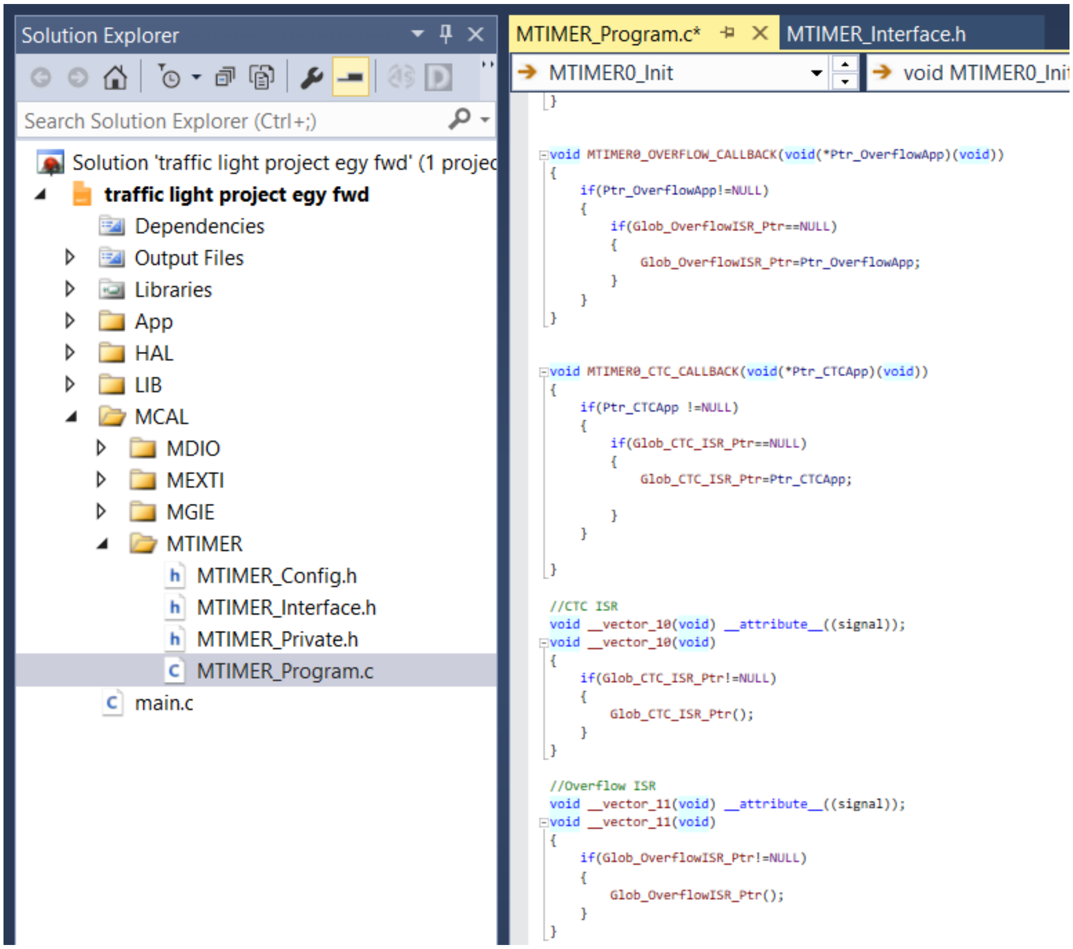
```c
#include <stdbool.h>
#include"../../LIB/STD_TYPES.h"
#include"../../LIB/BIT_MATH.h"
#include"../MDIO/MDIO_Interface.h"
#include"MTIMER_Config.h"
#include"MTIMER_Private.h"

//Two Global Pointers
void (*Glob_OverflowISR_Ptr)(void)=NULL;
void (*Glob_CTC_ISR_Ptr)(void)=NULL;

/*
 * Init:
 * 1-SetMode of Timer
 * 2-Set OC0 Mode
 * 3-Set Interrupt State
 * 4-Set Prescalar
 */

void MTIMER0_Init()
{
    //SetMode of Timer
#if TIMER0_MODE==TIMER_NORMAL_MODE
    CLR_BIT(TCCR0,3);
    CLR_BIT(TCCR0,6);

    //Set Normal Mode interrupt
    #if TIMER0_NORMAL_MODE_INTERRUPT_STATE==NORMAL_MODE_INTERRUPT_DISABLED
        CLR_BIT(TIMSK,0);
    #elif TIMER0_NORMAL_MODE_INTERRUPT_STATE==NORMAL_MODE_INTERRUPT_ENABLED
        SET_BIT(TIMSK,0);
    #endif

#elif TIMER0_MODE==TIMER_CTC_MODE
    SET_BIT(TCCR0,3);
    CLR_BIT(TCCR0,6);


    //Set OC0 Mode
    #if TIMER0_OC0_MODE==OC0_PIN_DISCONNECTED
        CLR_BIT(TCCR0,4);
        CLR_BIT(TCCR0,5);
    #elif TIMER0_OC0_MODE==OC0_PIN_TOGGLE
        SET_BIT(TCCR0,4);
        CLR_BIT(TCCR0,5);
    #elif TIMER0_OC0_MODE==OC0_PIN_CLEARED
        CLR_BIT(TCCR0,4);
        SET_BIT(TCCR0,5);
    #elif TIMER0_OC0_MODE==OC0_PIN_SET
        SET_BIT(TCCR0,4);
        SET_BIT(TCCR0,5);
    #endif
```



```
        //Set CTC Mode interrupt
    #if TIMER0_CTC_MODE_INTERRUPT_STATE==NORMAL_MODE_INTERRUPT_DISABLED
        CLR_BIT(TIMSK,1);
    #elif TIMER0_CTC_MODE_INTERRUPT_STATE==NORMAL_MODE_INTERRUPT_ENABLED
        SET_BIT(TIMSK,1);
    #endif

#elif TIMER0_MODE==TIMER_FAST_PWM_MODE
    SET_BIT(TCCR0,3);
    SET_BIT(TCCR0,6);

    //Set PWM Wave on OC0
    #if TIMER0_FAST_PWM_WAVE_MODE==FAST_PWM_OC0_DISCONNECTED
        CLR_BIT(TCCR0,4);
        CLR_BIT(TCCR0,5);
    #elif TIMER0_FAST_PWM_WAVE_MODE==FAST_PWM_OC0_NON_INVERTING
        CLR_BIT(TCCR0,4);
        SET_BIT(TCCR0,5);
    #elif TIMER0_FAST_PWM_WAVE_MODE==FAST_PWM_OC0_INVERTING
        SET_BIT(TCCR0,4);
        SET_BIT(TCCR0,5);
    #endif


#elif TIMER0_MODE==TIMER_PWM_PHASE_CORRECT_MODE
    CLR_BIT(TCCR0,3);
    SET_BIT(TCCR0,6);

    //Set PWM Wave on OC0
    #if TIMER0_PHASE_CORRECT_PWM_WAVE_MODE==PHASE_CORRECT_PWM_OC0_DISCONNECTED
        CLR_BIT(TCCR0,4);
        CLR_BIT(TCCR0,5);
    #elif TIMER0_PHASE_CORRECT_PWM_WAVE_MODE==PHASE_CORRECT_PWM_OC0_NON_INVERTING
        CLR_BIT(TCCR0,4);
        SET_BIT(TCCR0,5);
    #elif TIMER0_PHASE_CORRECT_PWM_WAVE_MODE==PHASE_CORRECT_PWM_OC0_INVERTING
        SET_BIT(TCCR0,4);
        SET_BIT(TCCR0,5);
    #endif

#else
#error "Wrong Mode chosen"
#endif
```



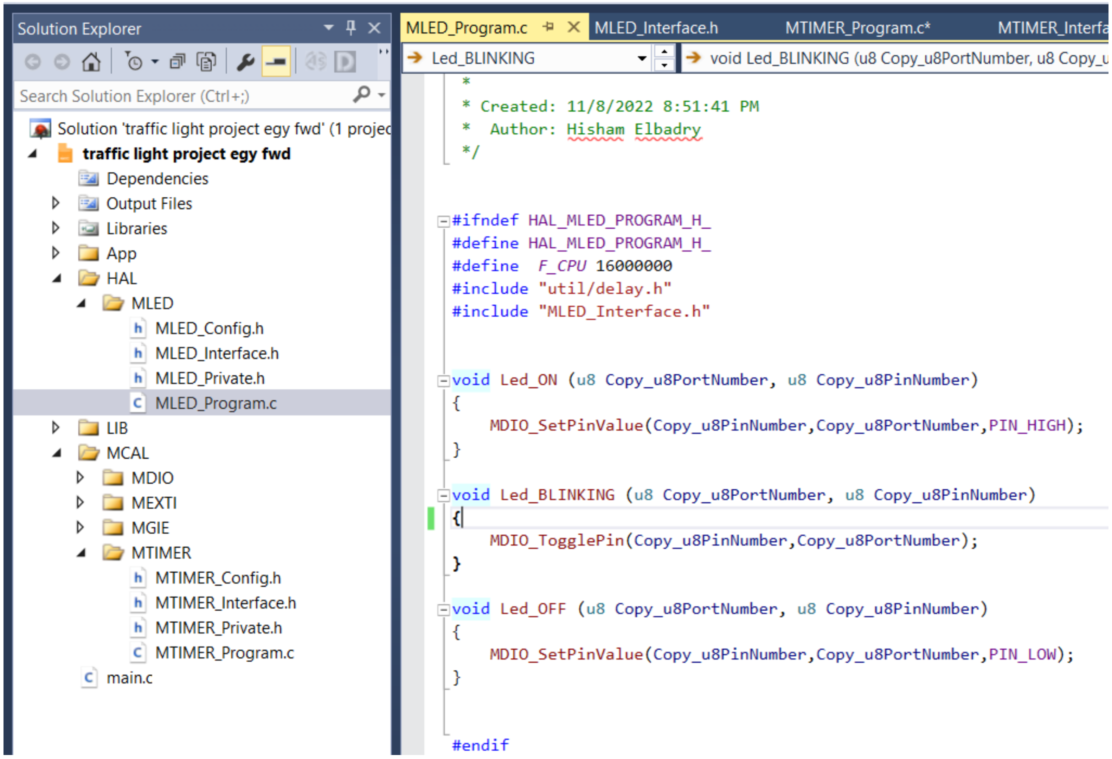
```
    //SET PRESCALAR
#if TIMER0_CLOCK_SOURCE==NO_CLOCK
    CLR_BIT(TCCR0,0);
    CLR_BIT(TCCR0,1);
    CLR_BIT(TCCR0,2);

#elif TIMER0_CLOCK_SOURCE==SYSTEM_CLOCK
    SET_BIT(TCCR0,0);
    CLR_BIT(TCCR0,1);
    CLR_BIT(TCCR0,2);

#elif TIMER0_CLOCK_SOURCE==PRESCALAR_OVER_8
    CLR_BIT(TCCR0,0);
    SET_BIT(TCCR0,1);
    CLR_BIT(TCCR0,2);

#elif TIMER0_CLOCK_SOURCE==PRESCALAR_OVER_64
    SET_BIT(TCCR0,0);
    SET_BIT(TCCR0,1);
    CLR_BIT(TCCR0,2);
#elif TIMER0_CLOCK_SOURCE==PRESCALAR_OVER_256
    CLR_BIT(TCCR0,0);
    CLR_BIT(TCCR0,1);
    SET_BIT(TCCR0,2);
#elif TIMER0_CLOCK_SOURCE==PRESCALAR_OVER_1024
    SET_BIT(TCCR0,0);
    CLR_BIT(TCCR0,1);
    SET_BIT(TCCR0,2);
#elif TIMER0_CLOCK_SOURCE==EXTERNAL_CLOCK_FALLING_EDGE
    CLR_BIT(TCCR0,0);
    SET_BIT(TCCR0,1);
    SET_BIT(TCCR0,2);
#elif TIMER0_CLOCK_SOURCE==EXTERNAL_CLOCK_RISING_EDGE
    SET_BIT(TCCR0,0);
    SET_BIT(TCCR0,1);
    SET_BIT(TCCR0,2);
#endif

}

void MTIMER0_SetPreload(u8 Copy_u8Preload)
{
    TCCR0=Copy_u8Preload;
}

void MTIMER0_SetCTC(u8 Copy_u8OCR)
{
    OCR0=Copy_u8OCR;
}
```



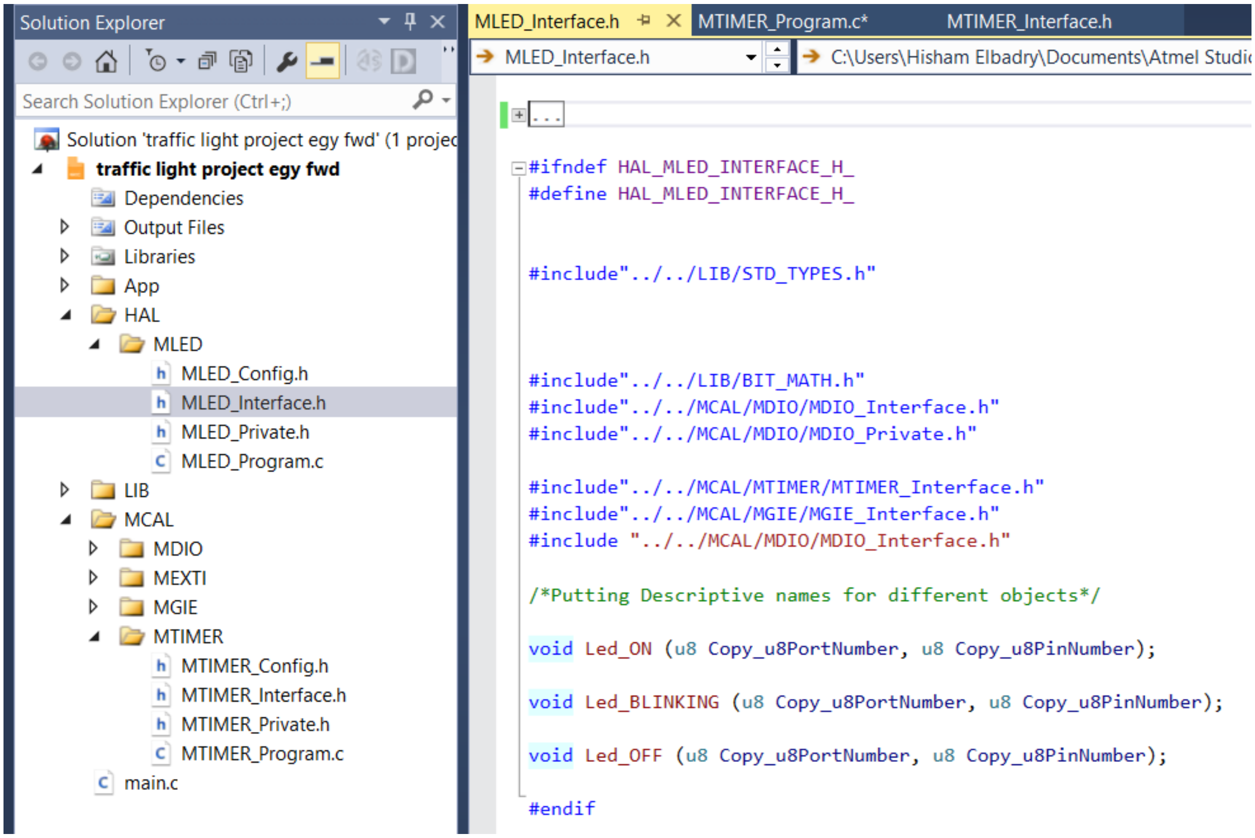
```
}


void MTIMER0_OVERFLOW_CALLBACK(void(*Ptr_OverflowApp)(void))
{
    if(Ptr_OverflowApp!=NULL)
    {
        if(Glob_OverflowISR_Ptr==NULL)
        {
            Glob_OverflowISR_Ptr=Ptr_OverflowApp;
        }
    }
}


void MTIMER0_CTC_CALLBACK(void(*Ptr_CTCApp)(void))
{
    if(Ptr_CTCApp !=NULL)
    {
        if(Glob_CTC_ISR_Ptr==NULL)
        {
            Glob_CTC_ISR_Ptr=Ptr_CTCApp;

        }
    }

}

//CTC ISR
void __vector_10(void) __attribute__((signal));
void __vector_10(void)
{
    if(Glob_CTC_ISR_Ptr!=NULL)
    {
        Glob_CTC_ISR_Ptr();
    }
}

//Overflow ISR
void __vector_11(void) __attribute__((signal));
void __vector_11(void)
{
    if(Glob_OverflowISR_Ptr!=NULL)
    {
        Glob_OverflowISR_Ptr();
    }
}
```

HAL files

LED files

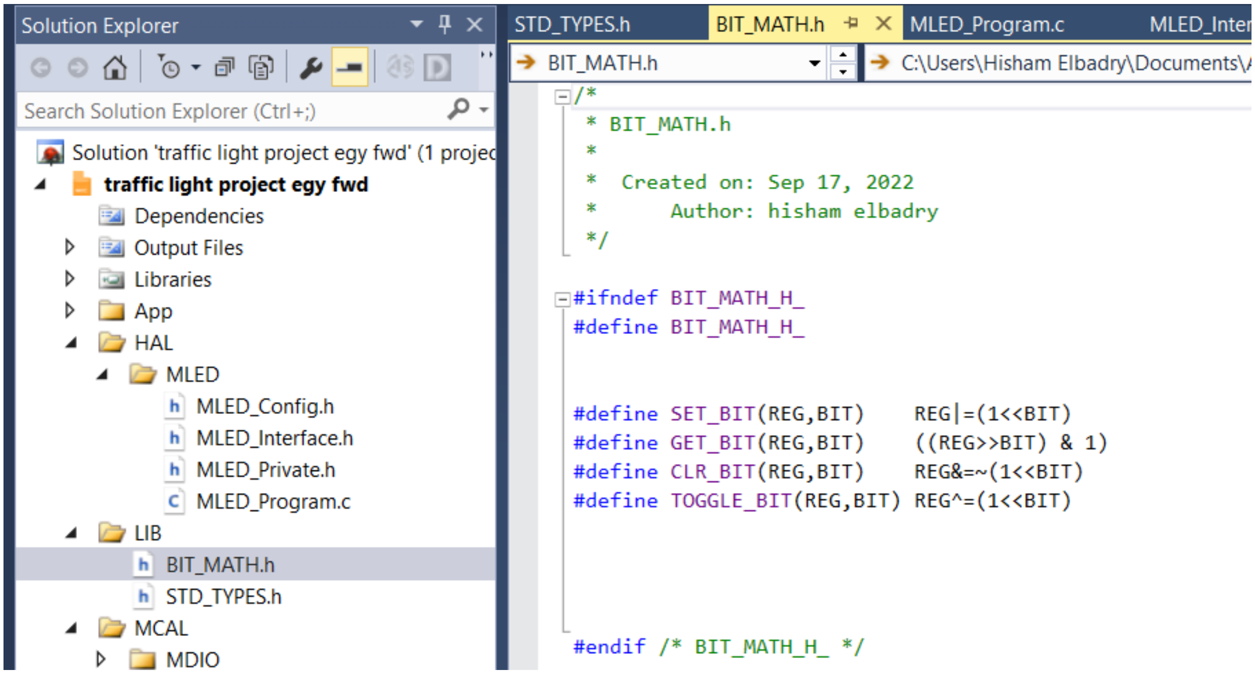


```
 *
 * Created: 11/8/2022 8:51:41 PM
 *  Author: Hisham Elbadry
 */


#ifndef HAL_MLED_PROGRAM_H_
#define HAL_MLED_PROGRAM_H_
#define  F_CPU 16000000
#include "util/delay.h"
#include "MLED_Interface.h"


void Led_ON (u8 Copy_u8PortNumber, u8 Copy_u8PinNumber)
{
    MDIO_SetPinValue(Copy_u8PinNumber,Copy_u8PortNumber,PIN_HIGH);
}

void Led_BLINKING (u8 Copy_u8PortNumber, u8 Copy_u8PinNumber)
{
    MDIO_TogglePin(Copy_u8PinNumber,Copy_u8PortNumber);
}

void Led_OFF (u8 Copy_u8PortNumber, u8 Copy_u8PinNumber)
{
    MDIO_SetPinValue(Copy_u8PinNumber,Copy_u8PortNumber,PIN_LOW);
}


#endif
```

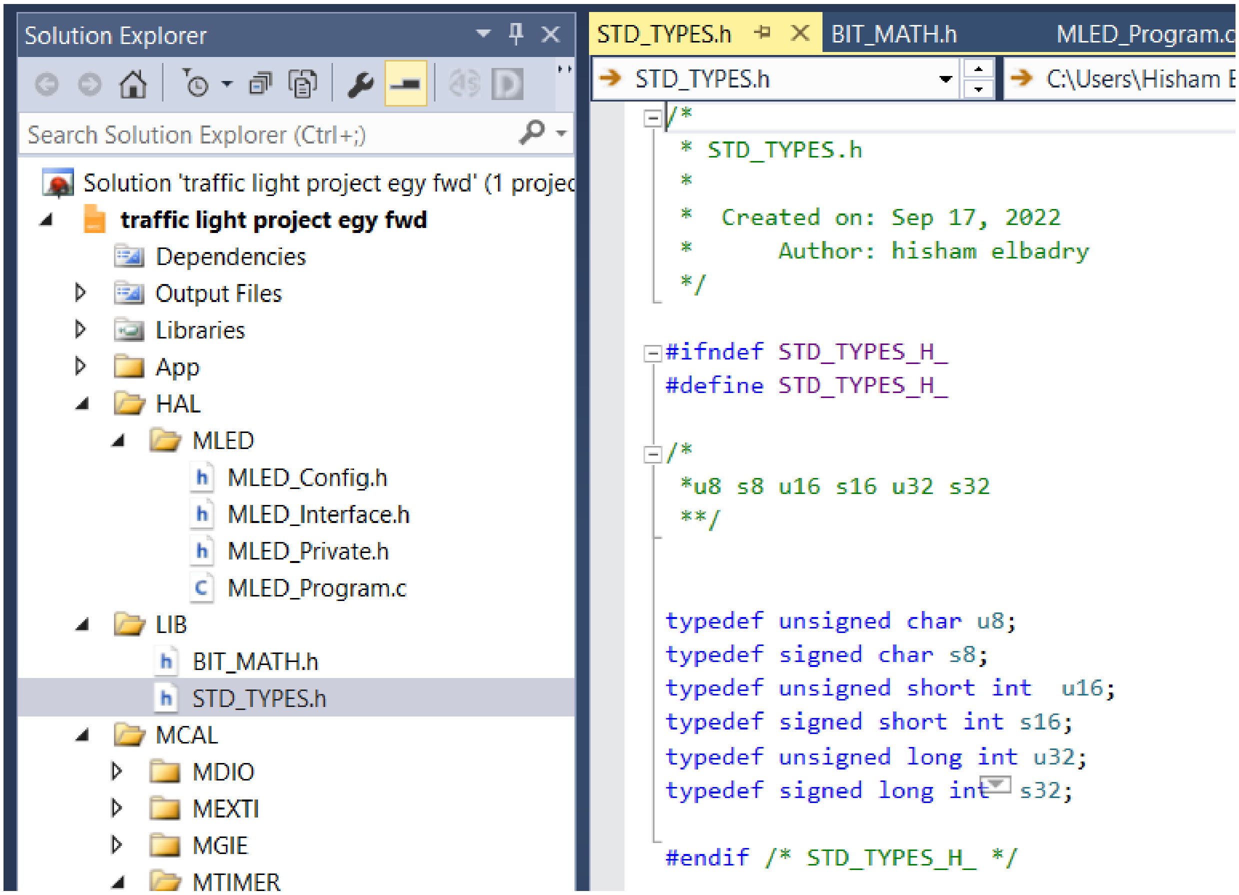


```
...

#ifndef HAL_MLED_INTERFACE_H_
#define HAL_MLED_INTERFACE_H_


#include"../../LIB/STD_TYPES.h"



#include"../../LIB/BIT_MATH.h"
#include"../../MCAL/MDIO/MDIO_Interface.h"
#include"../../MCAL/MDIO/MDIO_Private.h"

#include"../../MCAL/MTIMER/MTIMER_Interface.h"
#include"../../MCAL/MGIE/MGIE_Interface.h"
#include "../../MCAL/MDIO/MDIO_Interface.h"

/*Putting Descriptive names for different objects*/

void Led_ON (u8 Copy_u8PortNumber, u8 Copy_u8PinNumber);

void Led_BLINKING (u8 Copy_u8PortNumber, u8 Copy_u8PinNumber);

void Led_OFF (u8 Copy_u8PortNumber, u8 Copy_u8PinNumber);

#endif
```

Library files

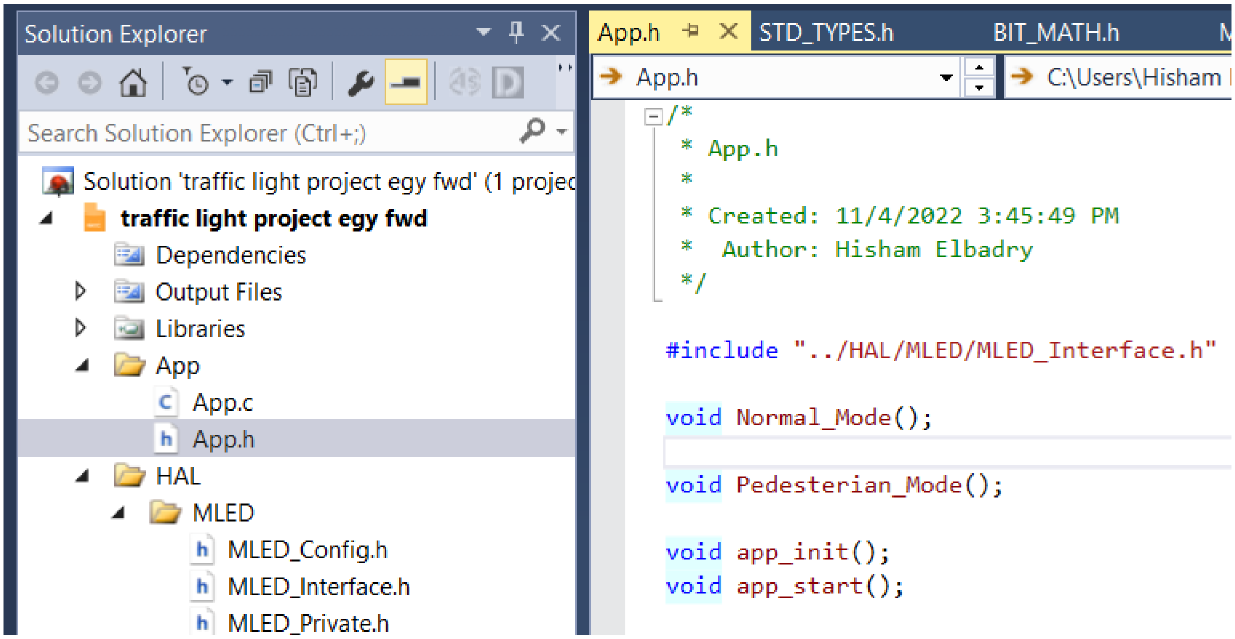


```
/*
 * BIT_MATH.h
 *
 *  Created on: Sep 17, 2022
 *      Author: hisham elbadry
 */

#ifndef BIT_MATH_H_
#define BIT_MATH_H_


#define SET_BIT(REG,BIT)    REG|=(1<<BIT)
#define GET_BIT(REG,BIT)    ((REG>>BIT) & 1)
#define CLR_BIT(REG,BIT)    REG&=~(1<<BIT)
#define TOGGLE_BIT(REG,BIT) REG^=(1<<BIT)




#endif /* BIT_MATH_H_ */
```

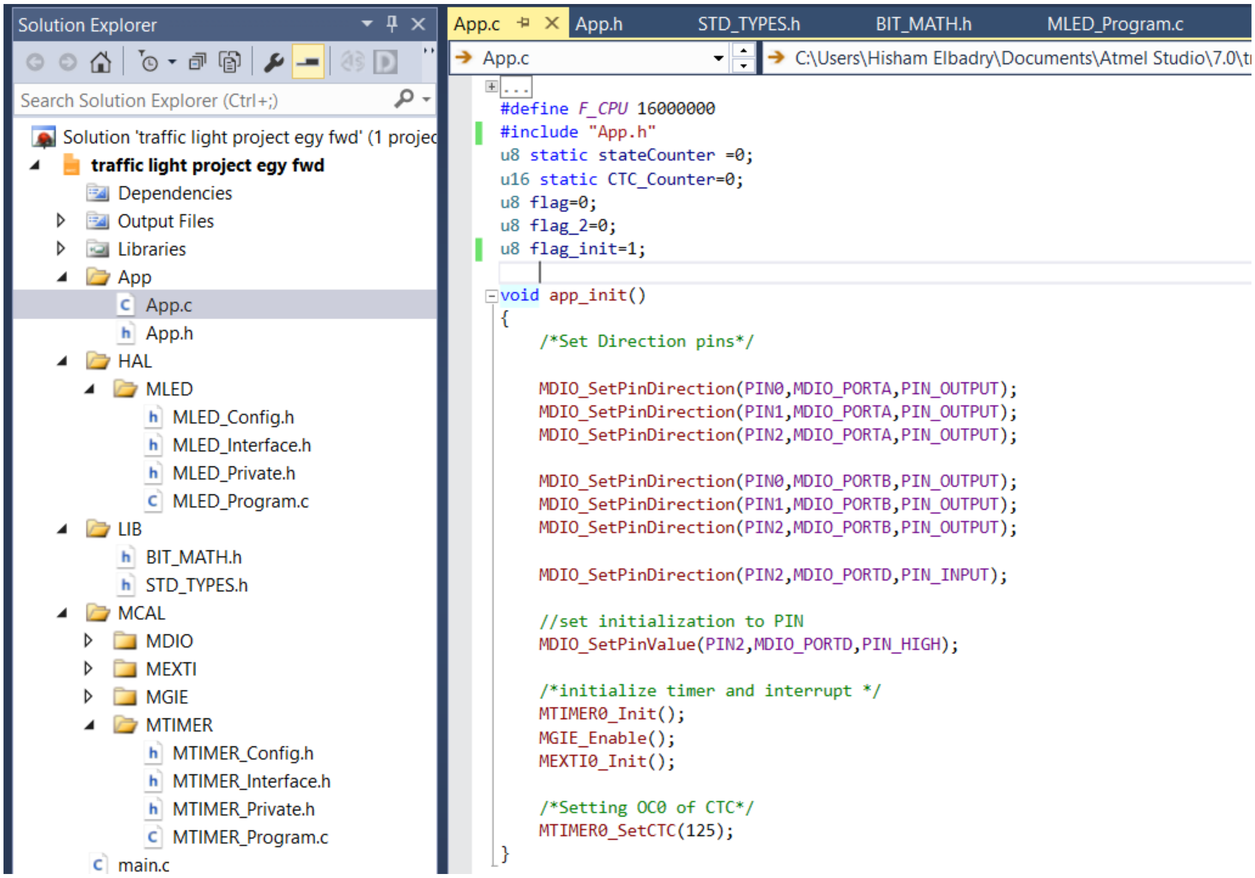


```
/*
 * STD_TYPES.h
 *
 *  Created on: Sep 17, 2022
 *      Author: hisham elbadry
 */

#ifndef STD_TYPES_H_
#define STD_TYPES_H_

/*
 *u8 s8 u16 s16 u32 s32
 **/


typedef unsigned char u8;
typedef signed char s8;
typedef unsigned short int  u16;
typedef signed short int s16;
typedef unsigned long int u32;
typedef signed long int s32;

#endif /* STD_TYPES_H_ */
```

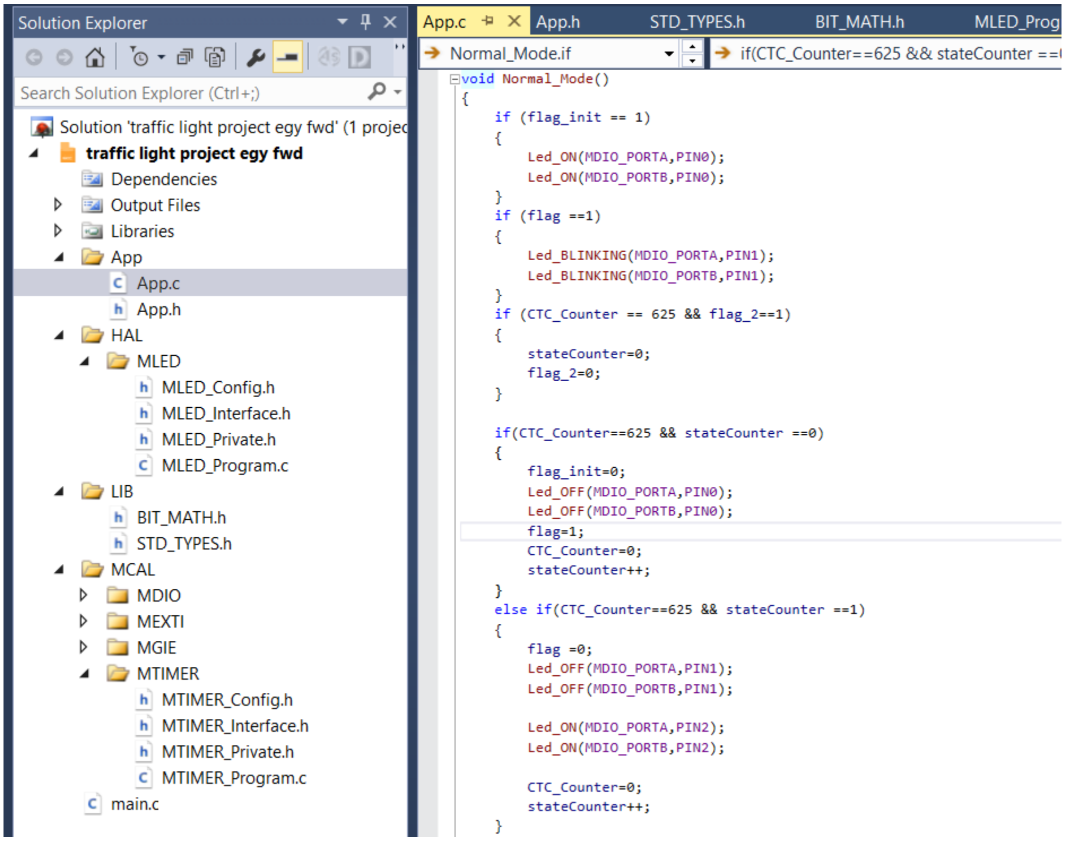
App files



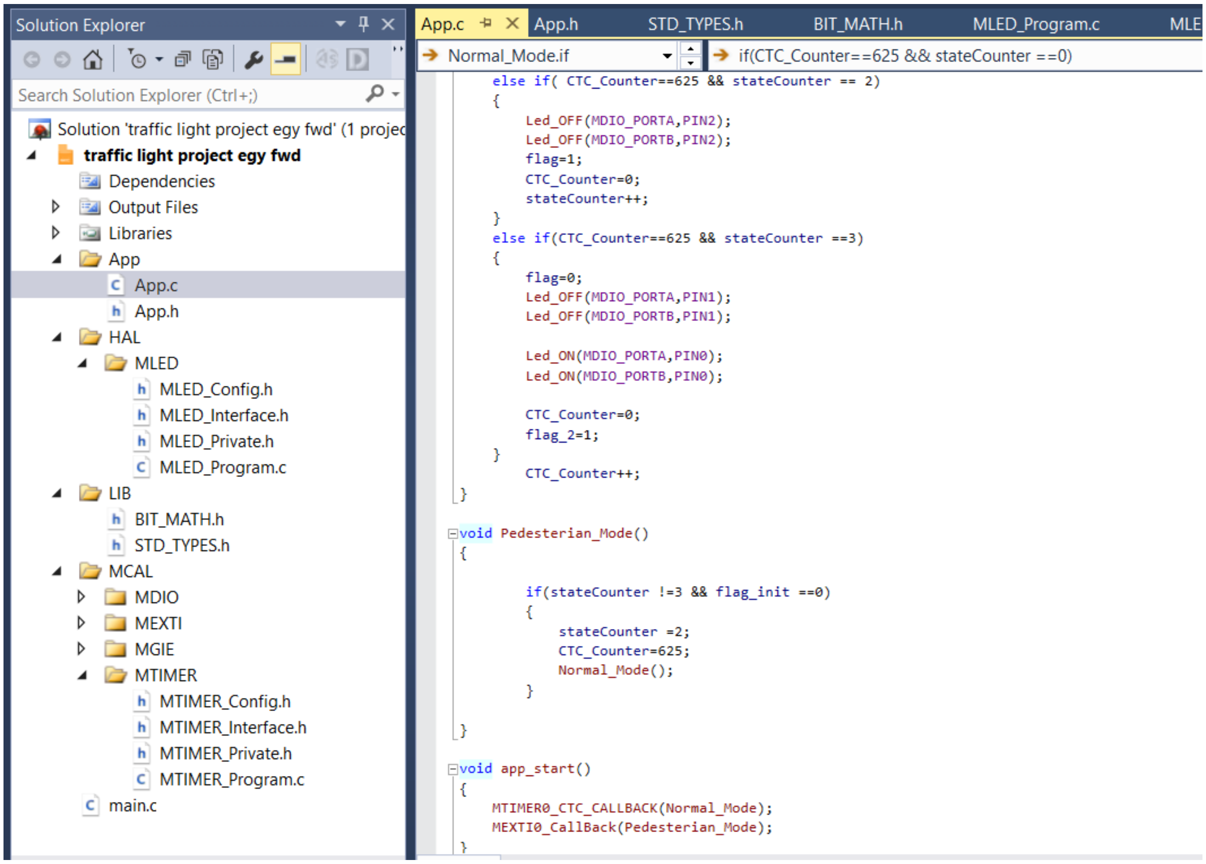
```c
/*
 * App.h
 *
 * Created: 11/4/2022 3:45:49 PM
 *  Author: Hisham Elbadry
 */

#include "../HAL/MLED/MLED_Interface.h"

void Normal_Mode();

void Pedesterian_Mode();

void app_init();
void app_start();
```



```c
...
#define F_CPU 16000000
#include "App.h"
u8 static stateCounter =0;
u16 static CTC_Counter=0;
u8 flag=0;
u8 flag_2=0;
u8 flag_init=1;

void app_init()
{
    /*Set Direction pins*/

    MDIO_SetPinDirection(PIN0,MDIO_PORTA,PIN_OUTPUT);
    MDIO_SetPinDirection(PIN1,MDIO_PORTA,PIN_OUTPUT);
    MDIO_SetPinDirection(PIN2,MDIO_PORTA,PIN_OUTPUT);

    MDIO_SetPinDirection(PIN0,MDIO_PORTB,PIN_OUTPUT);
    MDIO_SetPinDirection(PIN1,MDIO_PORTB,PIN_OUTPUT);
    MDIO_SetPinDirection(PIN2,MDIO_PORTB,PIN_OUTPUT);

    MDIO_SetPinDirection(PIN2,MDIO_PORTD,PIN_INPUT);

    //set initialization to PIN
    MDIO_SetPinValue(PIN2,MDIO_PORTD,PIN_HIGH);

    /*initialize timer and interrupt */
    MTIMER0_Init();
    MGIE_Enable();
    MEXTI0_Init();

    /*Setting OC0 of CTC*/
    MTIMER0_SetCTC(125);
}
```



```
void Normal_Mode()
{
    if (flag_init == 1)
    {
        Led_ON(MDIO_PORTA,PIN0);
        Led_ON(MDIO_PORTB,PIN0);
    }
    if (flag ==1)
    {
        Led_BLINKING(MDIO_PORTA,PIN1);
        Led_BLINKING(MDIO_PORTB,PIN1);
    }
    if (CTC_Counter == 625 && flag_2==1)
    {
        stateCounter=0;
        flag_2=0;
    }

    if(CTC_Counter==625 && stateCounter ==0)
    {
        flag_init=0;
        Led_OFF(MDIO_PORTA,PIN0);
        Led_OFF(MDIO_PORTB,PIN0);
        flag=1;
        CTC_Counter=0;
        stateCounter++;
    }
    else if(CTC_Counter==625 && stateCounter ==1)
    {
        flag =0;
        Led_OFF(MDIO_PORTA,PIN1);
        Led_OFF(MDIO_PORTB,PIN1);

        Led_ON(MDIO_PORTA,PIN2);
        Led_ON(MDIO_PORTB,PIN2);

        CTC_Counter=0;
        stateCounter++;
    }
```



```
    else if( CTC_Counter==625 && stateCounter == 2)
    {
        Led_OFF(MDIO_PORTA,PIN2);
        Led_OFF(MDIO_PORTB,PIN2);
        flag=1;
        CTC_Counter=0;
        stateCounter++;
    }
    else if(CTC_Counter==625 && stateCounter ==3)
    {
        flag=0;
        Led_OFF(MDIO_PORTA,PIN1);
        Led_OFF(MDIO_PORTB,PIN1);

        Led_ON(MDIO_PORTA,PIN0);
        Led_ON(MDIO_PORTB,PIN0);

        CTC_Counter=0;
        flag_2=1;
    }
        CTC_Counter++;
}

void Pedesterian_Mode()
{

        if(stateCounter !=3 && flag_init ==0)
        {
            stateCounter =2;
            CTC_Counter=625;
            Normal_Mode();
        }

}

void app_start()
{
    MTIMER0_CTC_CALLBACK(Normal_Mode);
    MEXTI0_CallBack(Pedesterian_Mode);
}
```