Table of Contents

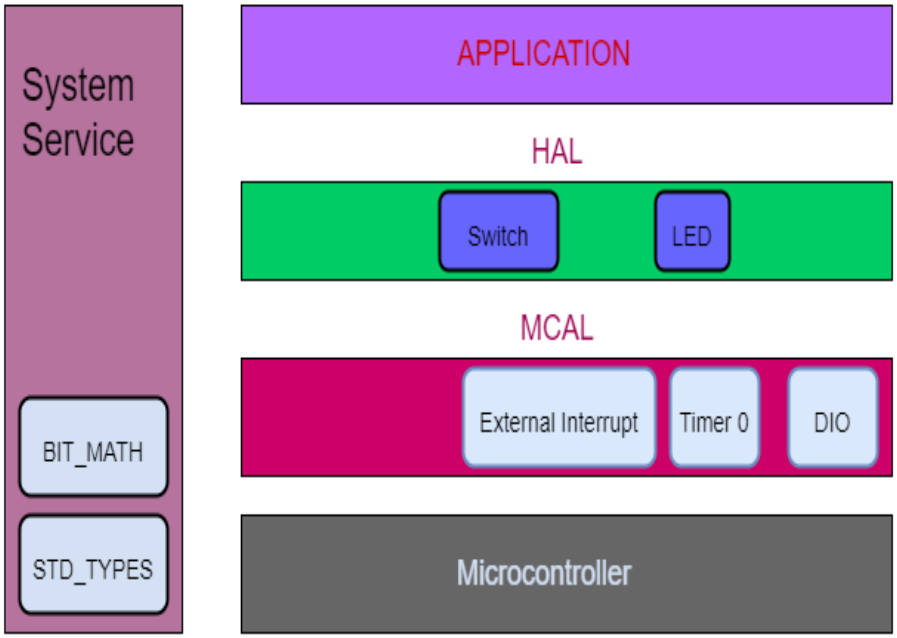
System Description

- 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

```c
#define OUTPUT 1

/* PIN Value Options */
#define HIGH   1
#define LOW    0
/**
 * @Function Name    :   DIO_SetPinValue
 * @description      :   This Function is used to write a value to a bit in I/O Registers
 * @param[in]        :   u8PortIdCopy , u8PinIdCopy , u8PinValCopy
 * @param[out]       :   EN_DioErrorStatus_t
 */
EN_DioErrorStatus_t DIO_SetPinValue(u8 u8PortIdCopy , u8 u8PinIdCopy, u8 u8PinValCopy);

/**
 * @Function Name    :   DIO_GetPinValue
 * @description      :   This Function is used to read a bit from I/O Registers
 * @param[in]        :   u8PortIdCopy , u8PinIdCopy , *u8PinValCopy--> Pointer to u8
 * @param[out]       :   EN_DioErrorStatus_t
 */
EN_DioErrorStatus_t DIO_GetPinValue(u8 u8PortIdCopy, u8 u8PinIdCopy, u8 *u8PinVal);

/**
 * @Function Name    :   DIO_SetPinDirection
 * @description      :   This Function is used to set the direction of a bit in I/O Registers
 * @param[in]        :   u8PortIdCopy , u8PinIdCopy , u8PinDirCopy
 * @param[out]       :   EN_DioErrorStatus_t
 */
EN_DioErrorStatus_t DIO_SetPinDirection (u8 u8PortIdCopy, u8 u8PinIdCopy, u8 u8PinDirCopy);
```

- 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

```c
/**
 *@Function Name    :   EXTI_voidInit
 *@description      :   This Function is used to set the configurations of the External Interrupt
 *@description      :   The Function is based on pre-build Configurations
 *@param[in]        :   void
 *@param[out]       :   void
 */
void EXTI_voidInit(void);
/**
 *@Function Name    :   EXTI_voidEnable
 *@description      :   This Function is used to set the configurations of the External Interrupt
 *@description      :   The Function is based on Run-time Configurations
 *@param[in]        :   EXTI_Config * Configurations  : pointer to struct that contains the Line number and the sense edge
 *@param[out]       :   void
 */
void EXTI_voidEnable(const EXTI_Config * Configurations);
/**
 *@Function Name    :   EXTI_voidDisable
 *@description      :   This Function is used to Disable specific External Interrupt Line
 *@param[in]        :   Copy_enumLine  :  External Interrupt Line number
 *@param[out]       :   void
 */
void EXTI_voidDisable(EXTI_Line Copy_enumLine);
/**
 *@Function Name    :   EXTI_voidSetCALLBACK
 *@description      :   This Function is used to set the Call back function of the External Interrupt Line
 *@param[in]        :   Copy_enumLine : External Interrupt Line number
 *@param[in]        :   void(*CALLBACK)(void) : Pointer to function
 *@param[out]       :   void
 */
void EXTI_voidSetCALLBACK(EXTI_Line Copy_enumLine, void(*CALLBACK)(void));
```

Writable | Smart Insert | 54 : 29

Type here to search | ENG | 1:40 PM 8/12/2022

- Global Interrupt

GIE_interface.h - Eclipse

AVR Run Window Help

APP.c | GIE_interface.h

```c
 * GIE_interface.h

#ifndef MCAL_INTERRUPT_GIE_INTERFACE_H_
#define MCAL_INTERRUPT_GIE_INTERFACE_H_

/**
 * @Function Name    :    GIE_voidEnable
 * @description      :    This Function is Used To enable the global Interrupt
 * @param[in]        :    void
 * @param[out]       :    void
 * */
void GIE_voidEnable(void);

/**
 * @Function Name    :    GIE_voidDisable
 * @description      :    This Function is Used To disable the global Interrupt
 * @param[in]        :    void
 * @param[out]       :    void
 * */
void GIE_voidDisable(void);

#endif /* MCAL_INTERRUPT_GIE_INTERFACE_H_ */
```

Writable | Smart Insert | 2 : 1

- Timer

01-AVR-WS - C/C++ - Traffic_Light/MCAL/TIMER0/TIMER0_interface.h - Eclipse

File Edit Source Refactor Navigate Search Project AVR Run Window Help

APP.c | TIMER0_interface.h | BUTTON_interface.h | LED_interface.h | STD_TYPES.h

```c
 * TIMER0_interface.h

#ifndef MCAL_TIMER0_TIMER0_INTERFACE_H_
#define MCAL_TIMER0_TIMER0_INTERFACE_H_

/**
 *@Function Name        :   TIM0_voidInit
 *@Function Description :   This Function is used to initialize the Timer0 Clock and prescaller and the Mode
 *@Function param[in]   :   void
 *@Function param[out]  :   void
 *@see                  :   TIMER0_config.h
 */
void TIM0_voidInit(void);
/**
 *@Function Name    :   TIM0_u8Set_CompareRegister_Value
 *@description      :   The Function is used to load the Compare register with a value
 *@param[in]        :   Copy_u8Counts  -> the Value to be loaded
 *@param[out]       :   void
 */
void TIM0_u8Set_CompareRegister_Value(u8 Copy_u8Counts);

/**
 *@Function Name    :   TIM0_voidSetCompareCallBack
 *@description      :   The Function is used to save the call back function of the Timer0 Compare Interrupt
 *@param[in]        :   void(*Copy_ptrCallBackFunc)(void)  -> pointer to function
 *@param[out]       :   u8 -> Error State
 */
u8 TIM0_voidSetCompareCallBack(void(*Copy_ptrCallBackFunc)(void));

#endif /* MCAL_TIMER0_TIMER0_INTERFACE_H_ */
```

Writable | Smart Insert | 31 : 42

Type here to search | ENG | 2:24 PM 8/12/2022

- LED

01-AVR-WS - C/C++ - Traffic_Light/HAL/LED/LED_interface.h - Eclipse

File Edit Source Refactor Navigate Search Project AVR Run Window Help

APP.c | BUTTON_interface.h | LED_interface.h | STD_TYPES.h

```c
/**
 * @Function Name    : LED_voidInit
 * @description      : This Function is used to initialize the LEDs as OUTPUT , Active_high/Active_Low
 *                   : The Function is based on the Link time Configurations
 * @see LED_config.c
 * @param[in]        : void
 * @param[out]       : EN_LedErrors_t -> Error state indicates the possible error may occur
 */

EN_LedErrors_t LED_voidInit ( void );


/**
 * @Function Name    : LED_voidON
 * @description      : This Function is used to Turn ON the LED
 * @see LED_config.c
 * @param[in]        : Copy_u8LedIndex  -> index of the LED in the array of the connected LEDs to MC
 * @param[out]       : EN_LedErrors_t -> Error state indicates the possible error may occur
 */
EN_LedErrors_t LED_voidON   ( u8 Copy_u8LedIndex);


/**
 * @Function Name    : LED_voidOFF
 * @description      : This Function is used to Turn OFF the LED
 * @see LED_config.c
 * @param[in]        : Copy_u8LedIndex  -> index of the LED in the array of the connected LEDs to MC
 * @param[out]       : EN_LedErrors_t -> Error state indicates the possible error may occur
 */
EN_LedErrors_t LED_voidOFF  ( u8 Copy_u8LedIndex);
```

Writable | Smart Insert | 48 : 1

Type here to search | ENG | 2:24 PM 8/12/2022

Push Button

01-AVR-WS - C/C++ - Traffic_Light/HAL/BUTTON/BUTTON_interface.h - Eclipse

File Edit Source Refactor Navigate Search Project AVR Run Window Help

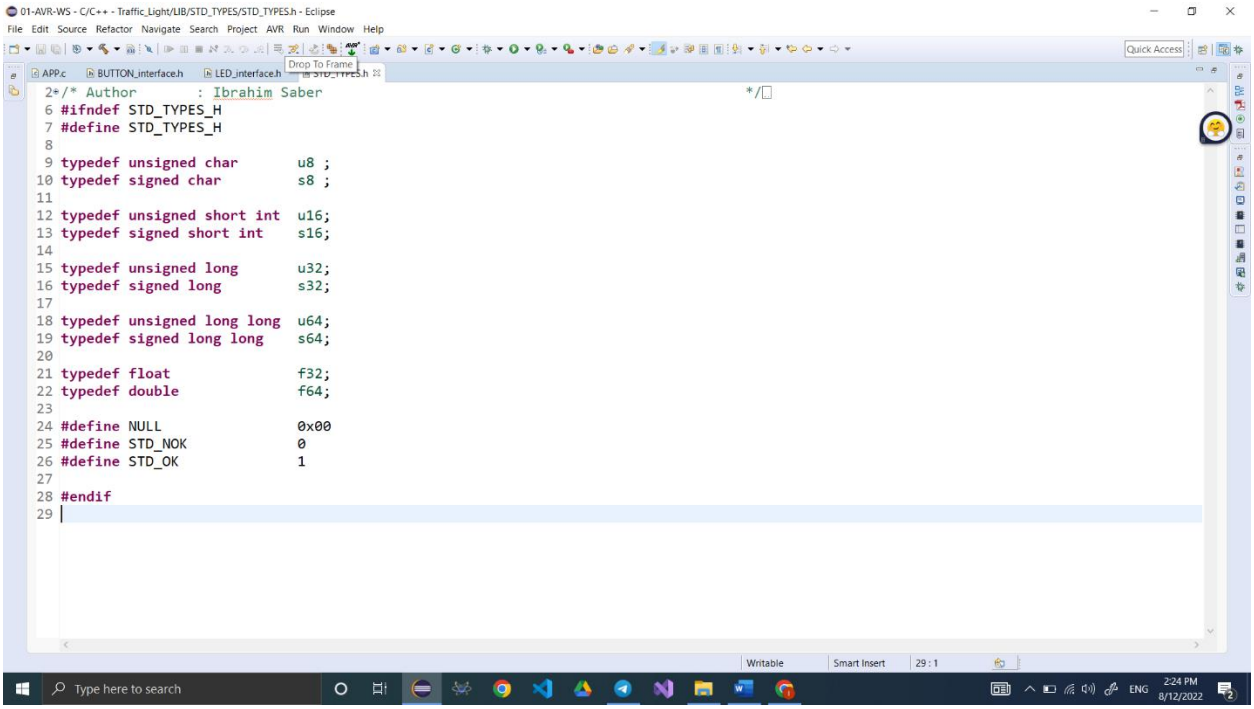
APP.c | BUTTON_interface.h | LED_interface.h | STD_TYPES.h

```c
/* Author        : Ibrahim saber                                    */

#ifndef BUTTON_INTERFACE_H
#define BUTTON_INTERFACE_H

/**
 * @Function Name    : BUTTON_voidInit
 * @description      : This Function is used to initialize the Buttons (INPUT , PULL_UP/PULL_DOWN)
 *                   : The function is based on Link time configurations
 * @param[in]        : void
 * @param[out]       : void
 */
void BUTTON_voidInit(void );

/**
 * @Function Name    : BUTTON_u8Read
 * @description      : This Function is used to get the Button value(HIGH / LOW)
 *                   : The function is based on Link time configurations
 * @param[in]        : Copy_u8ButtonIndex  -> index of the Button in the array of buttons connected to the MC
 * @param[out]       : u8 -> pressed value
 */
u8 BUTTON_u8Read    ( u8 Copy_u8ButtonIndex );

#endif
```

Writable | Smart Insert | 2 : 1

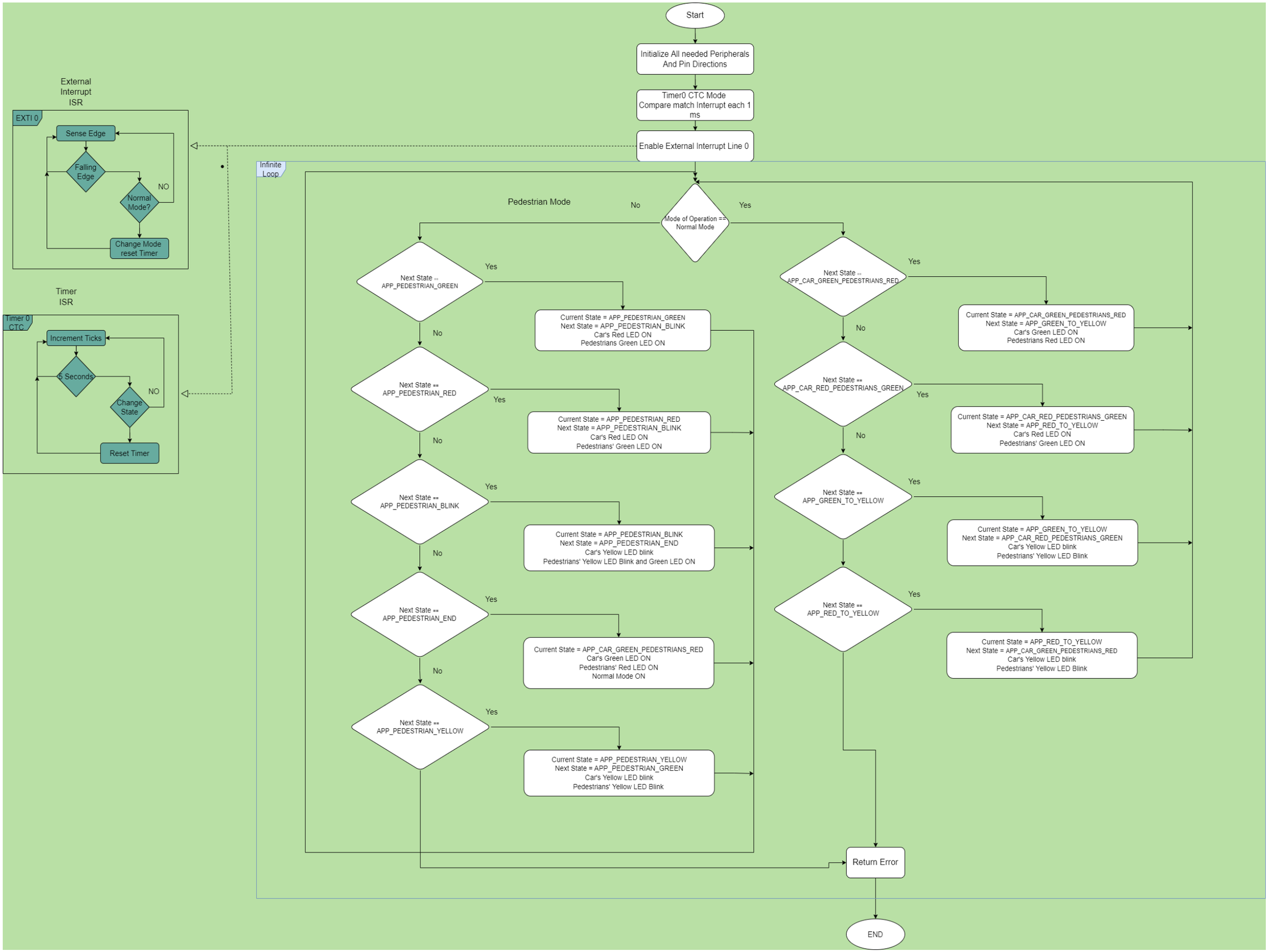
Type here to search | ENG | 2:24 PM 8/12/2022

Standard Types




```
/* Author        : Ibrahim Saber                                    */
#ifndef STD_TYPES_H
#define STD_TYPES_H

typedef unsigned char       u8 ;
typedef signed char         s8 ;

typedef unsigned short int  u16;
typedef signed short int    s16;

typedef unsigned long       u32;
typedef signed long         s32;

typedef unsigned long long  u64;
typedef signed long long    s64;

typedef float               f32;
typedef double              f64;

#define NULL                0x00
#define STD_NOK             0
#define STD_OK              1

#endif
```

System Flow Chart



System Constraints

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