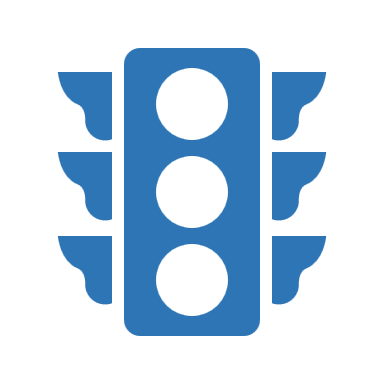
Traffic Light Control System Design



By: Ahmed Hassan Ibrahim

Table of Contents

[**System Description:** 3](#_Toc113997147)

[**State Machine:** 3](#_Toc113997148)

[**System Layers** 4](#_Toc113997149)

[**System Drivers** 4](#_Toc113997150)

[**DIO Driver:** 4](#_Toc113997151)

[Driver Functions: 4](#_Toc113997152)

[Driver Macros 5](#_Toc113997153)

[Driver Defines 5](#_Toc113997154)

[**Timer Driver:** 5](#_Toc113997155)

[Driver Functions: 5](#_Toc113997156)

[Driver Defines 5](#_Toc113997157)

[**LED Driver:** 6](#_Toc113997158)

[Driver Functions: 6](#_Toc113997159)

[**Button Driver:** 6](#_Toc113997160)

[Driver Functions: 6](#_Toc113997161)

[Driver Macros: 6](#_Toc113997162)

[**Interrupt Library:** 7](#_Toc113997163)

[Driver Defines 7](#_Toc113997164)

[**Utilities:** 7](#_Toc113997165)

[Registers defines: 7](#_Toc113997166)

[Types typedef: 8](#_Toc113997167)

[**Application:** 8](#_Toc113997168)

[Driver Functions: 8](#_Toc113997169)

[Application flags: 8](#_Toc113997170)

# **System Description:**

The system is designed to simulate traffic light system showing the flow of traffic light changes for both cars traffic light and pedestrian traffic light. The system actions change when the traffic light button is pressed by pedestrian depending on the state of the lights when pressed.

## **State Machine:**

Diagram

Description automatically generated

In the normal mode states colored blue the car lights change green – yellow – red each 5 seconds and the pedestrian traffic lights change opposite to that of car. For the yellow traffic light, it toggles each second. If the button is pressed while the car green or yellow light is on both the car and pedestrian yellow light will toggle and the pedestrian red light will be on at the same time for 5 seconds then the car red light will be on and pedestrian green light will be on allowing pedestrian to cross the road for 5 seconds then both yellow lights will toggle while pedestrian green light is still on for 5 seconds and return back to the normal state. If the button is pressed while car red is on nothing will happen as pedestrian can already cross the road.

# **System Layers**

The system is divided to 4 layers:

* Microcontroller
* MCAL
* ECUAL
* Application

# **System Drivers**

The system drivers that are used:

* DIO Driver
* Timer Driver
* Button Driver
* LED Driver
* Utilities for registers and types
* Interrupt library

Graphical user interface, chart

Description automatically generated

## **DIO Driver:**

### Driver Functions:

**void DIO\_init(uint8\_t pinNumber, uint8\_t portNumber, uint8\_t direction)**

* Initialize dio direction
* Output: High on DDR pinNumber

**void DIO\_write(uint8\_t pinNumber, uint8\_t portNumber, uint8\_t value)**

* Write data to dio
* Output: High on PIN pinNumber

**void DIO\_toggle(uint8\_t pinNumber, uint8\_t portNumber)**

* Toggle dio
* Output: toggle on PIN pinNumber

**void DIO\_read(uint8\_t pinNumber, uint8\_t portNumber, uint8\_t \*value)**

* Read dio
* Output: store the value of the pinNumber in value.

### Driver Macros

PORT\_A 🡪 'A'

PORT\_B 🡪 'B'

PORT\_C 🡪 'C'

PORT\_D 🡪 'D'

### Driver Defines

direction defines for DIO\_init function:

IN 🡪 0

OUT 🡪 1

value defines for DIO\_write function:

LOW 🡪 0

HIGH 🡪 1

## **Timer Driver:**

### Driver Functions:

**void TIMER\_init()**

* Initialize timer
* Output: low on TCCR0 and TCNT0

**void DELAY\_5\_sec ()**

* delay 5 seconds
* stopwatch = 5,000,000 us

**void DELAY\_1\_sec ()**

* delay 1 second
* stopwatch = 1,000,000 us

### Driver Defines

NUMBER\_OF\_OVERFLOW\_5 🡪 19531 (calculated overflow repeats for 5 second delay)

NUMBER\_OF\_OVERFLOW\_1 🡪 3906 (calculated overflow repeats for 1 second delay)

## **LED Driver:**

### Driver Functions:

**void LED\_init(uint8\_t ledPort, uint8\_t ledPin)**

* Initialize LED
* Output: High on DDR ledPin

**void LED\_on(uint8\_t ledPort, uint8\_t ledPin)**

* Turn on the LED
* Output: High on PIN ledPin

**void LED\_off(uint8\_t ledPort, uint8\_t ledPin)**

* Turn off the LED
* Output: low on PIN ledPin

**void LED\_toggle(uint8\_t ledPort, uint8\_t ledPin)**

* Toggle the LED
* Output: toggle PIN ledPin

## **Button Driver:**

### Driver Functions:

**void BUTTON\_init(uint8\_t buttonPort, uint8\_t buttonPin)**

* Initialize button
* Output: low on DDR buttonPin

**void BUTTON\_read(uint8\_t buttonPort, uint8\_t buttonPin, uint8\_t \*value)**

* Read button
* Output: store the value of the buttonPin in value.

### Driver Macros:

BUTTON\_1\_PORT 🡪 PORT\_D

BUTTON\_1\_PIN 🡪 2

LOW 🡪 0 (state Macros)

HIGH 🡪 1 (state Macros)

## **Interrupt Library:**

### Driver Defines

EXT\_INT\_0 \_\_vector\_1 (External Interrupt Requests 0)

EXT\_INT\_1 \_\_vector\_2 (External Interrupt Requests 1)

EXT\_INT\_2 \_\_vector\_3 (External Interrupt Requests 2)

sei() \_\_asm\_\_ \_\_volatile\_\_ ("sei" ::: "memory") (set global interrupts)

cli() \_\_asm\_\_ \_\_volatile\_\_ ("cli" ::: "memory") (clear global interrupts)

ISR(INT\_VECT)void INT\_VECT(void) \_\_attribute\_\_ ((signal,used));\

void INT\_VECT(void) (ISR definition)

## **Utilities:**

### Registers defines:

**(PIN A register)**

PORTA \*((volatile uint8\_t\*)0x3B)

DDRA \*((volatile uint8\_t\*)0x3A)

PINA \*((volatile uint8\_t\*)0x39)

**(PIN B register)**

PORTB \*((volatile uint8\_t\*)0x38)

DDRB \*((volatile uint8\_t\*)0x37)

PINB \*((volatile uint8\_t\*)0x36)

**(PIN C register)**

PORTC \*((volatile uint8\_t\*)0x35)

DDRC \*((volatile uint8\_t\*)0x34)

PINC \*((volatile uint8\_t\*)0x33)

**(PIN D register)**

PORTD \*((volatile uint8\_t\*)0x32)

DDRD \*((volatile uint8\_t\*)0x31)

PIND \*((volatile uint8\_t\*)0x30)

**(Timer registers)**

TCCR0 \*((volatile uint8\_t\*)0x53)

TCNT0 \*((volatile uint8\_t\*)0x52)

TIFR \*((volatile uint8\_t\*)0x58)

**(External interrupts registers)**

MCUCR \*((volatile uint8\_t\*)0x55)

MCUCSR \*((volatile uint8\_t\*)0x54)

GICR \*((volatile uint8\_t\*)0x5B)

GIFR \*((volatile uint8\_t\*)0x5A)

### Types typedef:

unsigned char uint8\_t (unsigned 8-bit character)

## **Application:**

### Driver Functions:

**void APP\_init()**

* Initialize all drivers
* Output: run all init functions

**void APP\_start()**

* Start the application
* Output: run the application

### Application flags:

unsigned int traffic\_status ( flag for button pushed at which state)

unsigned int interrupt\_flag (flag for when car red is on)

unsigned int yellow\_blink (flag to count number of times yellow light blinked)