<u>ON DEMAND TRAFFIC LIGHTS</u>

By: Shady Mohamed Abdelkahek

Table of Contents

1. System Description

This project is an on-demand traffic lights system. It consists of three LEDs for the cars, three LEDs for the pedestrians and a button for pedestrians which allows them to pass the road.

The system has two modes. The first mode, the system operates under normal conditions where there are no pedestrians and the traffic lights cycles through red, yellow and green then yellow again, where the yellow LED blinks. The second mode, the system operates when a pedestrian presses a button which allows them to pass the road depending on the stage at which the first mode was at. Then, after the second mode is executed, the system returns back to the first mode.

2. System Design

2.1. System Layers

The system has five layers. Application Layer, Utilities Layer, Electronic Unit Abstraction Layer (ECUAL), Microcontroller Abstraction Layer (MCAL) and Microcontroller Layer.

2.2. System Drivers

The system has four drivers. LED Driver, Button Driver, DIO Driver and Timer Driver. The DIO Driver and Timer Driver are located at the ECUAL Layer as well as Interrupt API. The LED Driver and Button Driver are located at the MCAL Layer. At the Utilities layer there are three APIs, bit manipulation, registers and types. At the Application Layer there is a driver that controls the logic of them system.

2.3. Drivers APIs

2.3.1. DIO API

```
#ifndef DIO_H_
#define DIO_H_
//include registers.h
#include "../../Utilities/registers.h"
#include "../../Utilities/bit_manipulation.h"
//all internal typedefs
//all driver macros
#define PORT_A 'A'
#define PORT_B 'B'
#define PORT_C 'C'
#define PORT_D 'D'
#define PIN0 0
#define PIN1 1
#define PIN2 2
#define PIN3 3
#define PIN4 4
#define PIN5 5
#define PIN6 6
#define PIN7 7
//Direction defines
#define IN 0
#define OUT 1
//Value defines
#define LOW 0
#define HIGH 1
//function prototypes
void DIO_init(uint8_t portNumber, uint8_t pinNumber, uint8_t direction); //Initialize DIO
Directions
void DIO_write(uint8_t portNumber, uint8_t pinNumber, uint8_t value);//Write Data to DIO
void DIO_toggle(uint8_t portNumber, uint8_t pinNumber);// Toggle DIO
void DIO_read(uint8_t portNumber, uint8_t pinNumber,uint8_t *value);// Read DIO

#endif /* DIO_H_ */
```

2.3.2. Interrupt API

```
#ifndef INTERRUPTS_H_
#define INTERRUPTS_H_

/* External Interrupt Request 0 */
#define EXT_INT_0 __vector_1
/* External Interrupt Request 1 */
#define EXT_INT_1 __vector_2
/* External Interrupt Request 2 */
#define EXT_INT_2 __vector_3

/*Set Global Interrupts, Set I-bit in SREG to 1 */
# define sei() __asm__ __volatile__ ("sei" ::: "memory");

/*Clear Global Interrupts, Set I-bit in SREG to 0 */
# define cli() __asm__ __volatile__ ("cli" ::: "memory");

/* ISR definition*/
#define ISR(INT_VECT)void INT_VECT(void) __attribute__ ((signal,used));\
void INT_VECT(void)



#endif /* INTERRUPTS_H_ */
```

2.3.3. Timer API

```
#ifndef TIMER_H_
#define TIMER_H_

#include "../../Utilities/registers.h"
#include "../../Utilities/bit_manipulation.h"
#include "../../Utilities/types.h"

#define  CPU_CLOCK 1 //CPU frequency Clock = 1MHz

extern uint8_t extinterrupt;

void timer_init(void);
void timer0_delay_ms(uint16_t delay);



#endif /* TIMER_H_ */
```

2.3.4. LED Driver API

```
#ifndef LED_H_
#define LED_H_
#include "../../MCAL/DIO Driver/dio.h"
#include "../../MCAL/Timer Driver/timer.h"

// Car Green LED Define
#define LED_CG_PORT PORT_A
#define LED_CG_PIN PIN0
// Car Yellow LED Define
#define LED_CY_PORT PORT_A
#define LED_CY_PIN PIN1
// Car Red LED Define
#define LED_CR_PORT PORT_A
#define LED_CR_PIN PIN2
//Pedestrian Green LED Define
#define LED_PG_PORT PORT_B
#define LED_PG_PIN PIN0
//Pedestrian Yellow LED Define
#define LED_PY_PORT PORT_B
#define LED_PY_PIN PIN1
//Pedestrian Red LED Define
#define LED_PR_PORT PORT_B
#define LED_PR_PIN PIN2
extern uint8_t extintblink;
void LED_init(uint8_t ledPort, uint8_t ledPin);
void LED_on(uint8_t ledPort, uint8_t ledPin);
void LED_off(uint8_t ledPort, uint8_t ledPin);
void LED_toggle(uint8_t ledPort, uint8_t ledPin);
void LED_blinkCar_5s(uint8_t ledPort, uint8_t ledPin,uint16_t delay, uint16_t
blinkfrequency);
void LED_blinkYellow_5s(uint8_t ledPort0, uint8_t ledPin0,uint8_t ledPort1, uint8_t
ledPin1,uint16_t delay,uint16_t blinkfrequency);

#endif /* LED_H_ */
```

2.3.5. Button API

```
#ifndef BUTTON_H_
#define BUTTON_H_
#include "../../MCAL/DIO Driver/dio.h"
#define EXTERNAL_INT_BUTTON_PORT PORT_D
#define EXTERNAL_INT_BUTTON_PIN PIN2
// Button State Macros
#define LOW 0
#define HIGH 1

//initialize
void BUTTON_init(uint8_t buttonPort,uint8_t buttonPin);


#endif /* BUTTON_H_ */
```

2.3.6. Application API

```c
#ifndef APPLICATION_H_
#define APPLICATION_H_

#include "../ECUAL/LED Driver/led.h"
#include "../ECUAL/Button Driver/button.h"
#include "../MCAL/Timer Driver/timer.h"
#include "../MCAL/Interrupt Library/interrupts.h"

void APP_init(void);
void APP_start(void);



#endif /* APPLICATION_H_ */
```

2.3.7. Bit Manipulation API

```c
#ifndef BIT_MANIPULATION_H_
#define BIT_MANIPULATION_H_

#define SET_BIT(x,bit) (x |=(1<<bit))
#define CLEAR_BIT(x,bit) (x &=~(1<<bit))
#define TOGGLE_BIT(x,bit) (x ^=(1<<bit))



#endif /* BIT_MANIPULATION_H_ */
```

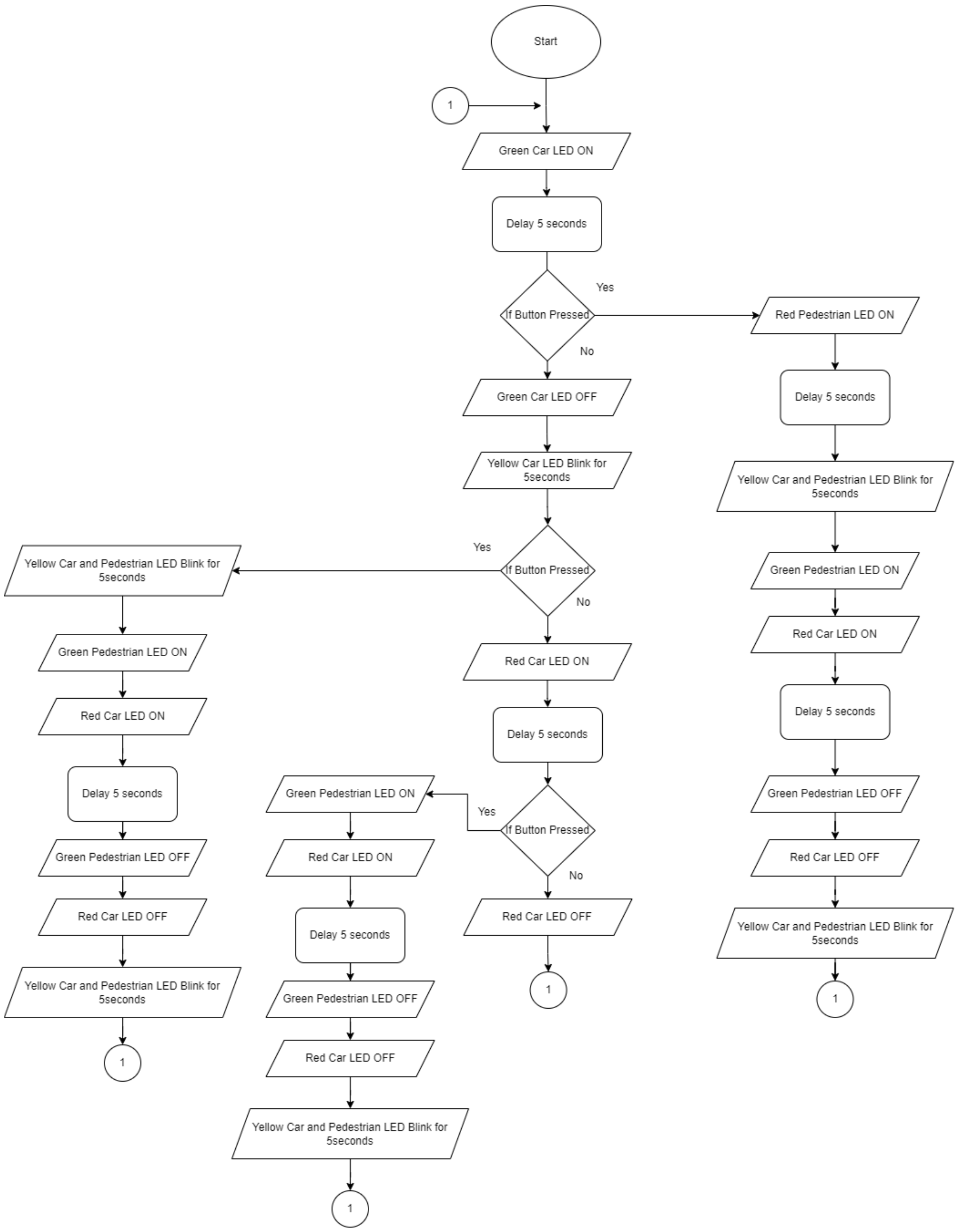
2.3.8. Registers API

```c
#ifndef REGISTERS_H_
#define REGISTERS_H_
#include "types.h"
/*************************************************************************/
/*  DIO Registers                                                        */
/*************************************************************************/

//PORTA Registers
#define PORTA *((volatile uint8_t*)0x3B) //8-bit register
#define DDRA *((volatile uint8_t*)0x3A) //8-bit register
#define PINA *((volatile uint8_t*)0x39) //8-bit register

//PORTB Registers
#define PORTB *((volatile uint8_t*)0x38) //8-bit register
#define DDRB *((volatile uint8_t*)0x37) //8-bit register
#define PINB *((volatile uint8_t*)0x36) //8-bit register

//PORTC Registers
#define PORTC *((volatile uint8_t*)0x35) //8-bit register
#define DDRC *((volatile uint8_t*)0x34) //8-bit register
#define PINC *((volatile uint8_t*)0x33) //8-bit register

//PORTD Registers
#define PORTD *((volatile uint8_t*)0x32) //8-bit register
#define DDRD *((volatile uint8_t*)0x31) //8-bit register
#define PIND *((volatile uint8_t*)0x30) //8-bit register

/*************************************************************************/
/*           Timer Registers                                            */
/*************************************************************************/
#define TCCR0 *((volatile uint8_t*)0x53) //Timer 0
#define TCNT0 *((volatile uint8_t*)0x52) //Timer 0
#define TIFR  *((volatile uint8_t*)0x58) //Timer 0 overflow flag

/*************************************************************************/
/*External Interrupt Registers                                           */
/*************************************************************************/
#define MCUCR *((volatile uint8_t*)0x55) //8-bit register
#define MCUCSR *((volatile uint8_t*)0x54) //8-bit register
#define GICR *((volatile uint8_t*)0x5B) //8-bit register
#define GIFR *((volatile uint8_t*)0x5A) //8-bit register

#endif /* REGISTERS_H_ */
```

2.3.9. Types API

```c
#ifndef TYPES_H_
#define TYPES_H_
typedef unsigned char uint8_t;
typedef unsigned short uint16_t;
typedef unsigned int uint32_t;
#endif /* TYPES_H_ */
```

3. System Flow Chart



4. System Constrains

This system has a constrain at which if the pedestrian button is pressed, no further action is taken if the button was pressed again. The button needs to only have a short press not a long press.