Embedded Systems
CESS Senior 1
Group 1 Section 1
TEAM 6
Final Project Documentation

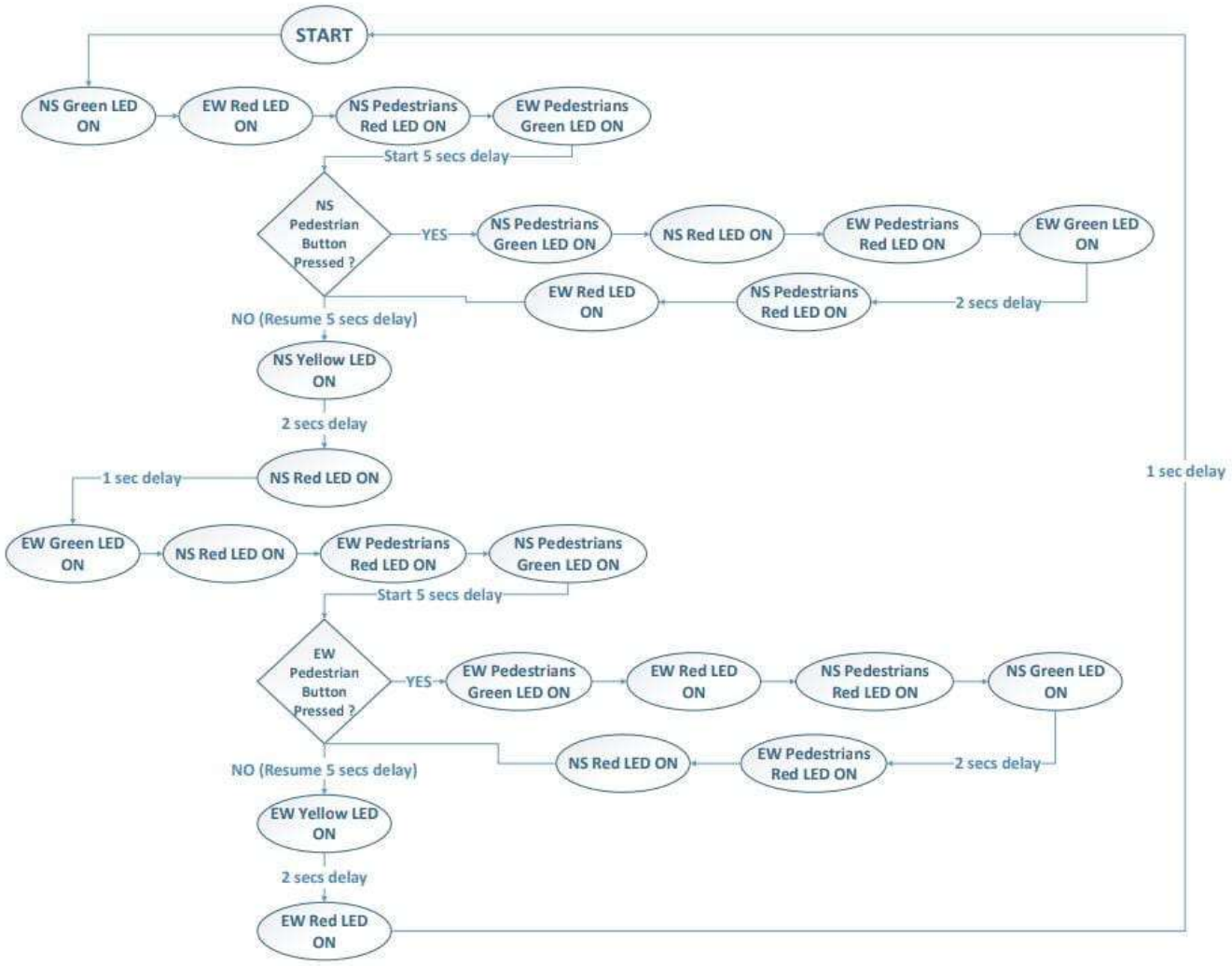
Abdelrahman Mahmoud Abdelrahman 18P6605
Shehab Mohamed Ibrahim 18P7213
Yusuf Sameh Fawzi 18P1399
Ahmad Ossama Ahmad 18P6575
Maryam Mohamed Abdelrahman 18P8171

Project Description

This project's purpose is to create a traffic light control system. There are just two traffic signals in the system. One permits automobiles to travel north to south, while the other allows them to travel east to west. A pedestrian traffic light is located next to each traffic signal. To cross the street safely, the pedestrian must touch a button to turn his signal green. Hence, we used 4 red LEDs, 4 green LEDs (for the 2 pedestrian and 2 normal traffic lights) and 2 yellow LEDs (for the 2 traffic lights)

Project Flowchart:

The flowchart will be found also as a pdf file on the drive link.



Our project code contains the following files:

- DIO.h
- DIO.c
- Timers.h
- Timers.c
- Types.h
- tm4c123gh6pm.h
- startup_ewarm.c
- main.c

DIO.h File Contents

1. Declaration of the following functions:

void PORTF_init(void);

void PORTB_init(void);

void PORTC_init(void);

void TrafficLEDs(bool NSR, bool NSY, bool NSG, bool EWR, bool EWY, bool EWG, bool PNSR, bool PNSG, bool PEWR, bool PEWG);

2. Enumeration of type State

It is created to include all the states that are available in the traffic system which are:

- NSGREEN which indicates NorthSouth green LED on
- NSYELLOW which indicates NorthSouth yellow LED on
- EWGREEN which indicates EastWest green LED on
- EWYELLOW which indicates EastWest yellow LED on
- switchToNS which indicates both lights red for 1 second, to switch to NorthSouth
- switchToEW which indicates both lights red for 1 second, to switch to EastWest

Sequence of traffic states:

NSGREEN → NSYELLOW → switchToEW → EWGREEN → EWYELLOW → switchToNS

As the North south traffic starts with green for 5 seconds then turns yellow for 2 seconds then turn red and through 1 second delay switchToEW occurs where all traffic lights are red then East west traffic turns from red to green (EWGREEN) and continue green for 5 seconds then turns yellow for 2 seconds (EWYELLOW) then turn red and through 1 second delay (switchToNS) occurs where all traffic lights are red then switched to (NSGREEN) state and the TrafficLeds function determines these colors through its parameters. This sequence is applied also for the pedestrian traffic opposite to its corresponding normal as the pedestrian can pass the street when the opposite normal traffic is green.

DIO.c File Contents

Functions Implementation:

void TrafficLEDs (bool NSR, bool NSY, bool NSG, bool EWR, bool EWY, bool EWG, bool PNSR, bool PNSG, bool PEWR, bool PEWG):

The parameters represent the LEDs of each traffic is on or off (1 or 0).

Interrupts are disabled to prevent unexpected behavior before lighting the LEDs

Then each Port Data pin in port B is connected with its corresponding normal traffic LED as shown in the code

And each Port Data pin in port C is connected with its corresponding pedestrian traffic LED as shown in the code

Then Interrupts are enabled again in case a pedestrian button is pressed or a timer delay is finished.

These symbols refer to the state of each normal traffic light:

- NSR = NorthSouth Red
- NSY = NorthSouth Yellow
- NSG = NorthSouth Green
- EWR = EastWest Red
- EWY = EastWest Yellow
- EWG = EastWest Green

These symbols refer to the state of each pedestrian traffic light:

- PNSR = Pedestrians NorthSouth Red
- PNSG = Pedestrians NorthSouth Green
- PEWR = Pedestrians EastWest Red
- PEWG = Pedestrians EastWest Green

The parameters are Boolean where 1 means the led light is on and 0 means the led light is off

PORTF_init():

Initialization of port F to be able to use the switches 1 & 2 of Tiva (pin 0 & pin 4) to turn on the pedestrian traffic

Steps of initialization:

1. configure the system clock
2. check the system clock is configured after passing specific number of clock cycles
3. The port F is unlocked using this value 0x4C4F434B
4. GPIO_PORTC_CR_R = 0x1F to allow changes to the 5 pins only (0 --> 4)
5. Make direction of the switches input (0) and LEDs output (1) which is (01110 I binary) → (0x0E)
6. Configure the switches pins to be pull up resistor (0x11) using GPIO_PORTF_PUR_R
7. Port F is enabled (all 5 switches are on) so it is written (11111 in binary) → (1F in Hexadecimal)
8. GPIO_PORTF_IEV_R sets the switches' pins to 1 to activate interrupts when switch is pressed (negative edge triggered)
9. interrupt priority is set to 3 using NVIC_PRI7_R
10. Interrupt Clear Register set the switches' pins to 1 to clear the interrupt if the flag (RIS) is up
11. GPIO_PORTF_IM_R set the switches' pins to 1 to enable interrupts for both switches
12. NVIC_EN0_R register is set to enable all the interrupts

PORTB_init():

Initialization of port B function is called to be able to use the 6 LEDs of the normal traffic where PORT B Data pins are connected to LEDs as follow:

- NorthSouth Red connected to pin 0
- NorthSouth Yellow connected to pin 1
- NorthSouth Green connected to pin 2
- EastWest Red connected to pin 3
- EastWest Yellow connected to pin 4
- EastWest Green connected to pin 7

Steps of initialization:

1. configure the system clock
2. check the system clock is configured after passing specific number of clock cycles
3. GPIO_PORTC_CR_R = 0x9F to allow changes to these pins only (0 --> 4 and 7)
4. As shown above from 0 --> 4 pins and pin 7 are output LEDs which are set to (1 in binary) so GPIO_PORTB_DIR_R equals (10011111) → (0x9F) in register
5. GPIO_PORTC_PUR_R 0x00 as no pull up resistor in Port B
6. GPIO_PORTB_DEN_R is the same as the direction register as we are enabling all the pins used

PORTC_init():

Initialization of port C function is called to be able to turn on the 4 LEDs of the pedestrian traffic where PORT C Data pins are connected to LEDs as follow:

- Pedestrians NorthSouth Red connected to pin 4
- Pedestrians NorthSouth Green connected to pin 5
- Pedestrians EastWest Red connected to pin 6
- Pedestrians EastWest Green connected to pin 7

Steps of initialization:

1. configure the system clock
2. check the system clock is configured after passing specific number of clock cycles
3. GPIO_PORTC_CR_R = 0xF0 to allow changes to these pins only (4 --> 7)
4. As shown above from 4 --> 7 pins are output LEDs which are set to (1 in binary) so GPIO_PORTB_DIR_R equals (11110000) → (0xF0) in register
5. GPIO_PORTC_PUR_R 0x00 as no pull up register in Port C
6. GPIO_PORTB_DEN_R is the same as the direction register as we are enabling all the pins used

void PortFHandler(void):

This function does not take any parameters, it handles the pedestrians' interrupts by checking the switch button pushed, if switch1 is pushed then the interrupt comes from East West traffic and if switch 2 is pressed then it is the North South traffic. Then it calls the TrafficLEDs function to turn on red light for normal traffic and turn on green light for pedestrian traffic and turn the green light for the opposite normal traffic. Then Timer1delay is called to start timer for 2 seconds, and after 2 seconds pass the timer1delay is exited and the interrupt flag is cleared to return to normal state. This function is already declared by the startup_ewarm.c file. This function is implemented but not called anywhere in the code as the processor executes it once an interrupt occurs so that it can handle that interrupt

Timer.h File Contents

The declarations of the following functions:

void TIMER0_init(void);

void TIMER1_init(void);

void TIMER0_delay(uint32 delay);

void TIMER1_delay(uint32 delay);

Timer.c File Contents

Timer.h file is included which contains the timers' functions declaration

TIMER0_init():

Timer0 is initialized to count the timer of each normal traffic until an interrupt occurs

TIMER1_init():

Timer1 is initialized to count the timer of each pedestrian traffic once a pedestrian button is pushed

void TIMER0_delay (uint32 delay):

This function parameter takes the time duration of the supposed traffic light in milliseconds and put this value in the TIMER0_TAILR_R and start counting down while checking for the value of TIMER0_TAILR_R to be equal zero which means the timer is finished not interrupt from pedestrian button as whenever the timer finishes counting it calls the timer0handler function which reset the TIMER0_TAILR_R value so the code exits the function TIMER0_delay and state is switched

Initialization steps:

1. We enable the system clock
2. Disable timer to avoid unexpected behavior (cannot write on an enabled timer)
3. Configure 32-bit timer using one shot mode
4. No pre-scalar is used (Frequency / 1)
5. Time out flag is cleared
6. Interrupts in enabled from the mask register
7. Interrupt priority set to 4 which has lower priority than the interrupt of timer1 as cars have to stop and pedestrian have the advantage to pass when presses the switch
8. NVIC_EN0_R enables interrupt number 19

void TIMER1_delay (uint32 delay):

This function parameter takes the time duration of the supposed pedestrian light (2000) in milliseconds and put this value in the TIMER0_TAILR_R and start counting down and wait for interrupt as whenever the timer finishes counting it calls the timer1handler function which disables the timer so the code exits the function TIMER1_delay and returns back to its normal state.

Timer1_delay has the same initialization steps as Timer0_delay except the interrupt priority is set to 2 and NVIC_EN0_R enables interrupt number 21 which has higher priority than the interrupt of timer0 as pedestrian have the advantage to pass when presses the switch

The following functions are timers' handlers that are implemented but not called anywhere in the code as the processor executes them once an interrupt occurs so that they can handle that interrupt.

void TIMER0Handler (uint32 delay):

The only purpose of this function is to stop the timer when the time of the lighting led is reached so it makes TIMER0_TAILR_R = 0 so the timer0delay function knows time is up. Then it set TIMER0_ICR_R = 1 to clear the RIS flag

void TIMER1Handler (uint32 delay):

This handler stops the timer after the 2 seconds delay is finished and clear the interrupt flag using TIMER1_ICR_R

- Types.h contains some defined datatypes
- tm4c123gh6pm.h contains all registers stored in Tiva
- startup_ewarm.c contains the declaration of handlers

(void Timer0Handler(void) & void Timer1Handler(void) & void PortFHandler(void))

These handlers are not called; however, the processor execute them when an interrupt occurs in order to handle that interrupt as mentioned earlier.

main.c File Contents

All ports initializations are called and all interrupts are enabled

volatile enumeration TrafficState of type state is initialized with NSGREEN which is the first state of the sequence

void RestoreState(void):

RestoreState does not take any parameters, it has a switch case that takes the TrafficState and determine the led that should light up. It does that by calling TrafficLeds function which turn on/off all the LEDs in the 2 normal traffic lights and 2 pedestrian traffic lights according to the state of the traffic. It is called continuously when timer0 of normal traffic is counting. If Timer 0 is interrupted at any given time (pedestrian is passing the street), so when the pedestrians interrupt is served, the system should be able to restore its state (traffic normal condition).

At each switch case the LEDs are lighten according to the sequence shown above.

This is the order of states in the TrafficLeds function:

(NSR, NSY, NSG, bool EWR, bool EWY, bool EWG, bool PNSR, bool PNSG, bool PEWR, PEWG)

For example, at the north south normal traffic green and yellow states the south east pedestrian will be green as the south east normal traffic at this time will be red and the north south pedestrian traffic will be red of course and vice versa.

During the switchtoNS and switchtoEW states all the traffic and pedestrians red lights will be on to get ready for the next state as their delays are only 1 second.