RMIT University

School of Engineering

EEET2096 – Embedded Systems Design and Implementation

Lab Experiment #3

C Compiler and Traffic Lights

Lecturer: Dr Jidong Wang

Tutor: Tuan Phan

Students: Jaime De Esteban Uranga s3762844,

Jasmine Rehal s3541157, Traratuch Wattanatorn s3610249

(Identical Report)

Group: Tuesday 12:30 – 14:30

Submission Due Date: Sunday 05/05/19 23:59

Introduction:

Laboratory 3 was integrated around executing a state machine code utilising C code. In the tasks, the LEDs were utilised in order to generate a traffic light controller. A state machine concept is used in the laboratory tasks. In a state machine system, the next state depends on the current state plus the inputs. The input-output on the ARM board will be utilised to simulate the lights as well as the sensors at a traffic intersection. The main objectives of laboratory 3 are to perform code written in C language in the Kiel Integrated Development Environment, to become trained in writing code in C language in order to monitor as well as control the input-output for the ARM processor and to develop a state machine that utilises C language code. The laboratory will also use code supplied to present messages on the LCD that is provided on the ARM board. The design tasks were to create a simplified intersection traffic light controller with right turn sensors installed. It was assumed that each direction has a straight light has only two possible colours as well as a right turn arrow light having only 2 possible colours [1].

Preliminary:

1. ***Use the code of Lab2 appendix as a guide to give the C code to initialise GPIOE to drive the LEDs. Note initialisation will include enabling the clock. The header file <stm32f10x_cl.h> containing all the definitions of the registers assumes structures. Therefore the code will be of the form:***
   For the registers structures code is:

```
RCC->APB2ENR |= 1 << 6; // Enable GPIOE clock
RCC->APB2ENR |= 1 << 5; //Enable GPIOD clock
GPIOE->CRH = 0x33333333; //Configure the GPIO for LEDs
```

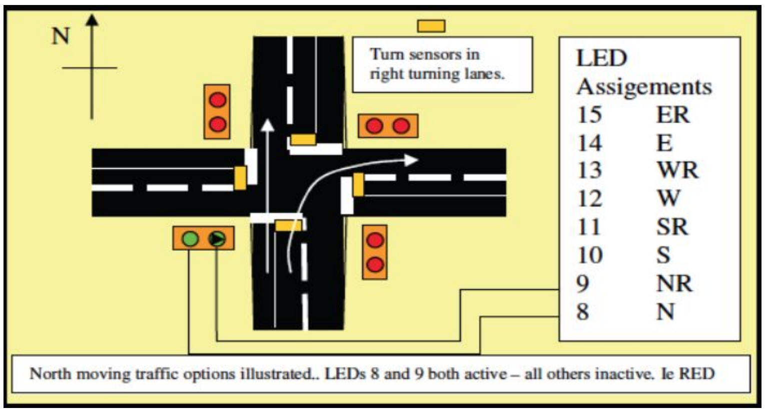
2. ***Use a C code to implement a delay as a function. You can adjust the delay length via a function parameter. You can call this function whenever you need a delay. A delay function with name Read_joystick( ) is given in the appendix. The function not only gives a delay, but also reads the joystick.***
   For implement for delay function is `DELAY_C 0x30000` and this function put on reads joystick as a part of delay function.

```
void waitF(int nOfUnits){
 int countWait=nOfUnits;
        countWait*=DELAY_C;

        while(countWait!=0){
                joystick=Read_joystick();
                countWait-=0x1;
```

Procedure:

In this lab working on C code program to executes in loop included delay routine to order the traffic light (by turn on and off) using joystick as a turn sensore to order function by pushing up,down,left,right for requise the green lights. By follow design tasks to enable `CRH` for LED.


N
Turn sensors in right turning lanes.
LED Assigements
15 ER
14 E
13 WR
12 W
11 SR
10 S
9 NR
8 N
North moving traffic options illustrated.. LEDs 8 and 9 both active – all others inactive. Ie RED

Questions:

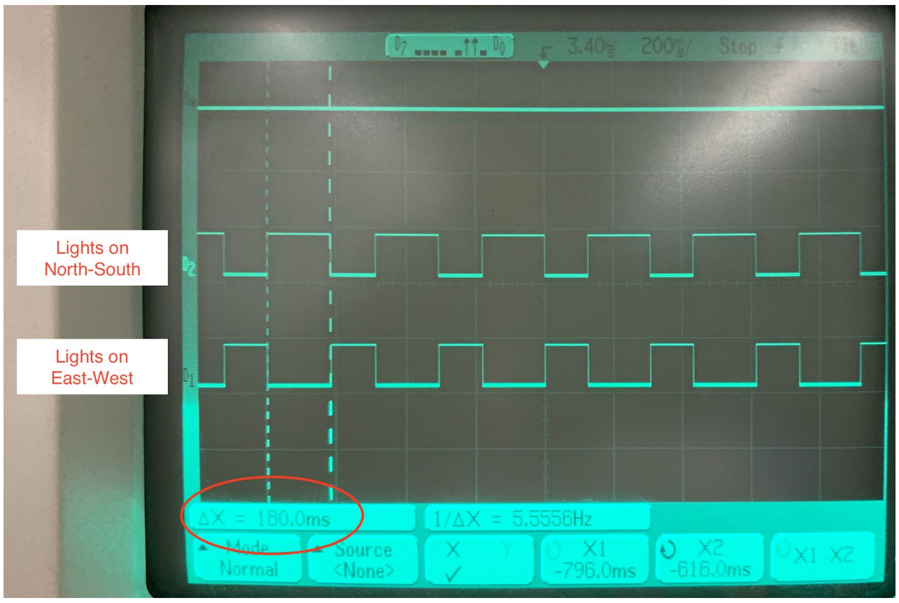
Question 1: In the sample/simplified start up code given in figure 16.24 what do the directives/pseudo instructions EXPORT and IMPORT do?

In the sample/simplified start up code that is given in figure 16.24, the directives/pseudo instructions are

Question 2: The ARM processor has the potential to operate lighting fast. Ignoring the delay routine what determines the time in each state, what is the approximate time to perform the code in each state and what is the fastest speed this design can operate.

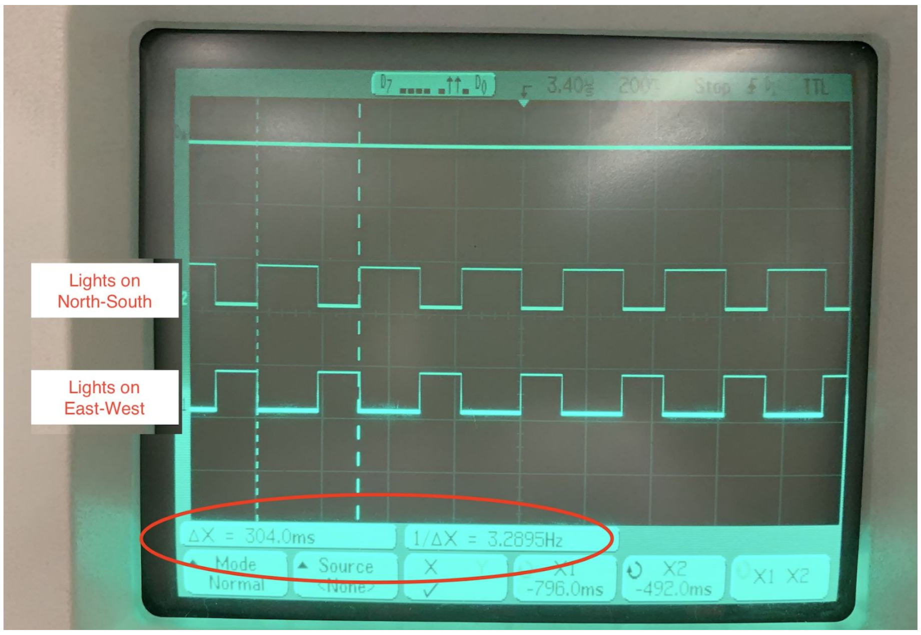
Ignoring the delay routine, the aspect that determines the time in each state is the display changing text, the leds turning on and off and the lines of code running without any delay. Although without a delay function the joystick will never be read because there is no time to be sampled.

The approximate time to perform the code in each state using the MSO we can approximate it to be:



We can estimate the average time in a state to be 180ms.

The fastest speed that this design can operate is dependant on how fast the board can change through states. The period of the loop is given by the MSO and looking at the trace we can estimate that the period is 304ms and this the maximum speed / frequency of the design switching between states is 3.29Hz



Code Explanation::

```
int Read_joystick(){
int stick;
stick = joystick;
stick |=~GPIOD->IDR; //read joystick each loop
stick &= 0xf000; //relevant bits
return stick;
}

void waitF(int nOfUnits){
 int countWait=nOfUnits;
     countWait*=DELAY_C;

     while(countWait!=0){
          joystick=Read_joystick();
          countWait-=0x1;
     }
}
```

We have modified the given code to work under a wait function ("waitF") that accepts an integer as the number of units that are needed to wait. For example, during normal use, the lights are running for 10 time units so what the code does is call waitF(10) and then the function multiplies 10 times a specified delay "DELAY_C" and starts a countdown.

While this countdown is running, we call another function called Read_Joystick() that returns an int that updates our Joystick reading. The rest of this function is the same as the given code but instead of having a loop inside its a simple function that grabs the reading from the input.

In our case we have specified 4 "boolean" type of variables to record if a right turn has happened and if so, we need to "reset" the joystick to accept another turn.

We have also implemented the extra functionality to make the traffic light system behave like a more realistic implementation of a real traffic light system.

Therefore, when the system detects that there is only one right turn requested, and the opposite hasn't been triggered, the light will be green for the right turn and the straight crossing, alleviating any traffic that is waiting to cross the intersection.

A diagram of that scenario is shown below, in Figure 1, which depicts when a right turn sensor is triggered on the North-Right section but the South-Right sensor has NOT been triggered, allowing for traffic to flow in the directions of North-Right as well as North-South.

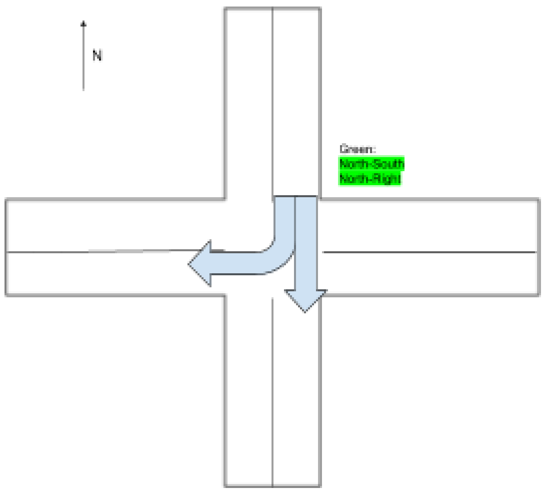


Figure 1. Diagram showing situation when right turn sensor is triggered on the North-right section but on the South-right section.

***Question 3:** Why do the response times to a request vary and with your design what is the worst case response to a request? It will probably be necessary to include your state machine design as part of your answer.*

Response times vary depending on when the request (when the sensor detects a car/when the user presses the joystick), because if the press is registered right before the “if” statement, this enables the traffic light to enable that right turn, and hence, the response time will be very good.
Inversely, if the button press is registered when the traffic lights are red and green for the other side of the intersection, the response will have to wait until the lights turn red for that side and then the if statement to will be evaluated. The following figures, shown below show the state machine design for both the best case scenario and the worst case scenario for the west turn request, respectively.

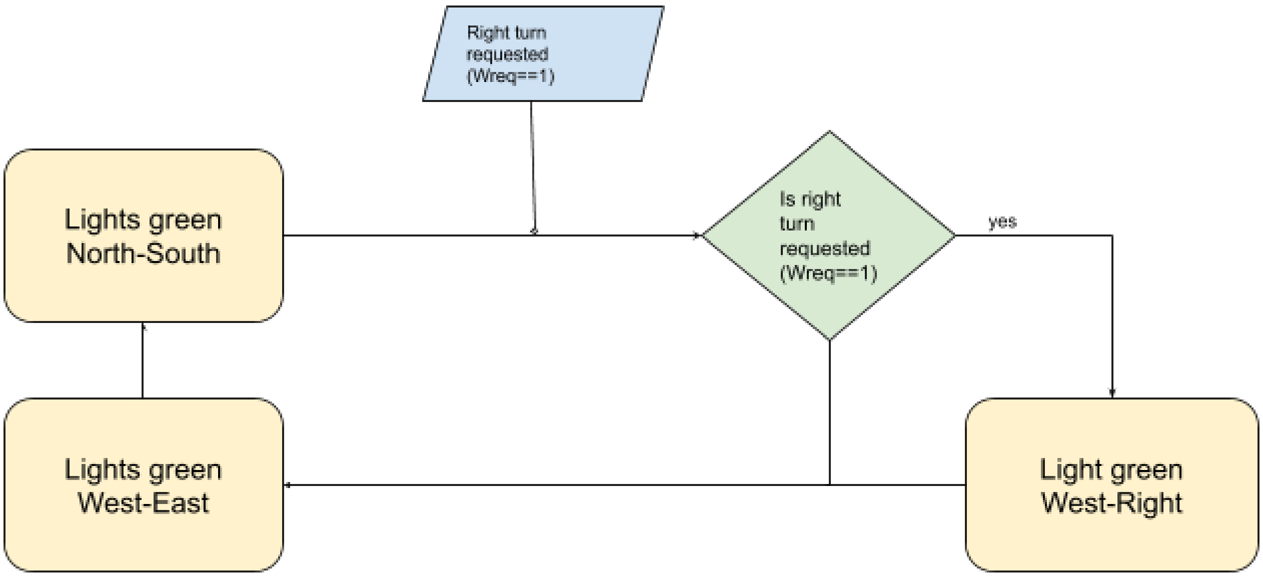


Figure 2. State Machine design for best case scenario for West Turn Request

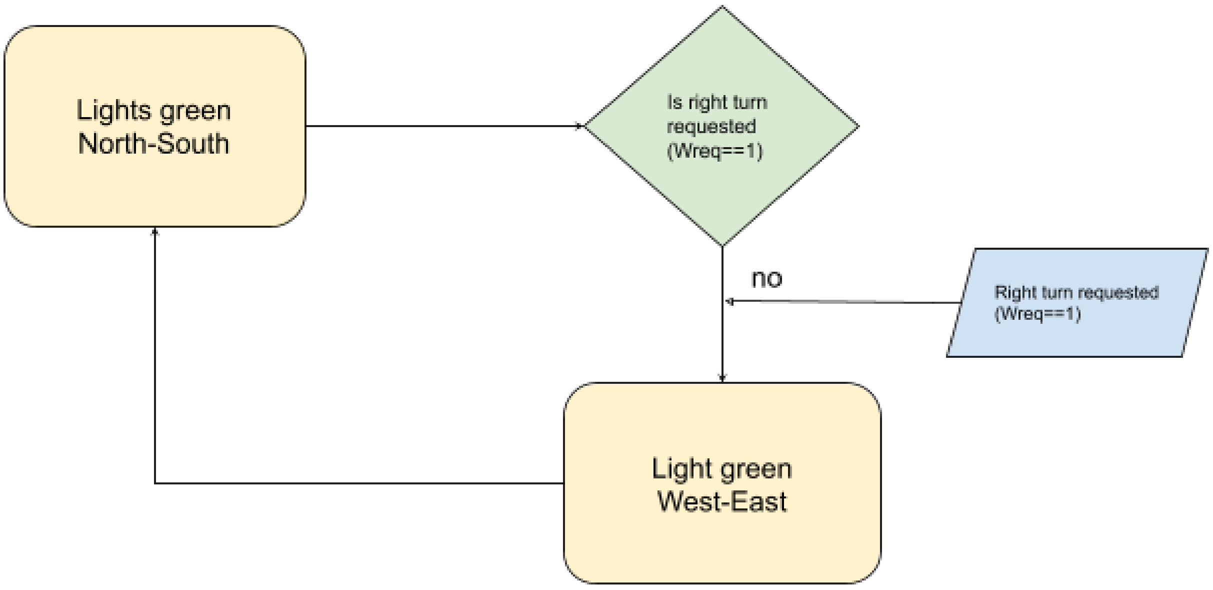

Lights green North-South
Is right turn requested (Wreq==1)
no
Right turn requested (Wreq==1)
Light green West-East

Figure 3. State Machine design for worst case scenario for West Turn Request

Figure 4, shown below depicts the MSO trace of the program running normally, the pins that are shown here represent the lights for North and for West. As you can see, every 10 time units, they both alternate, which lets traffic go through each side of the intersection.

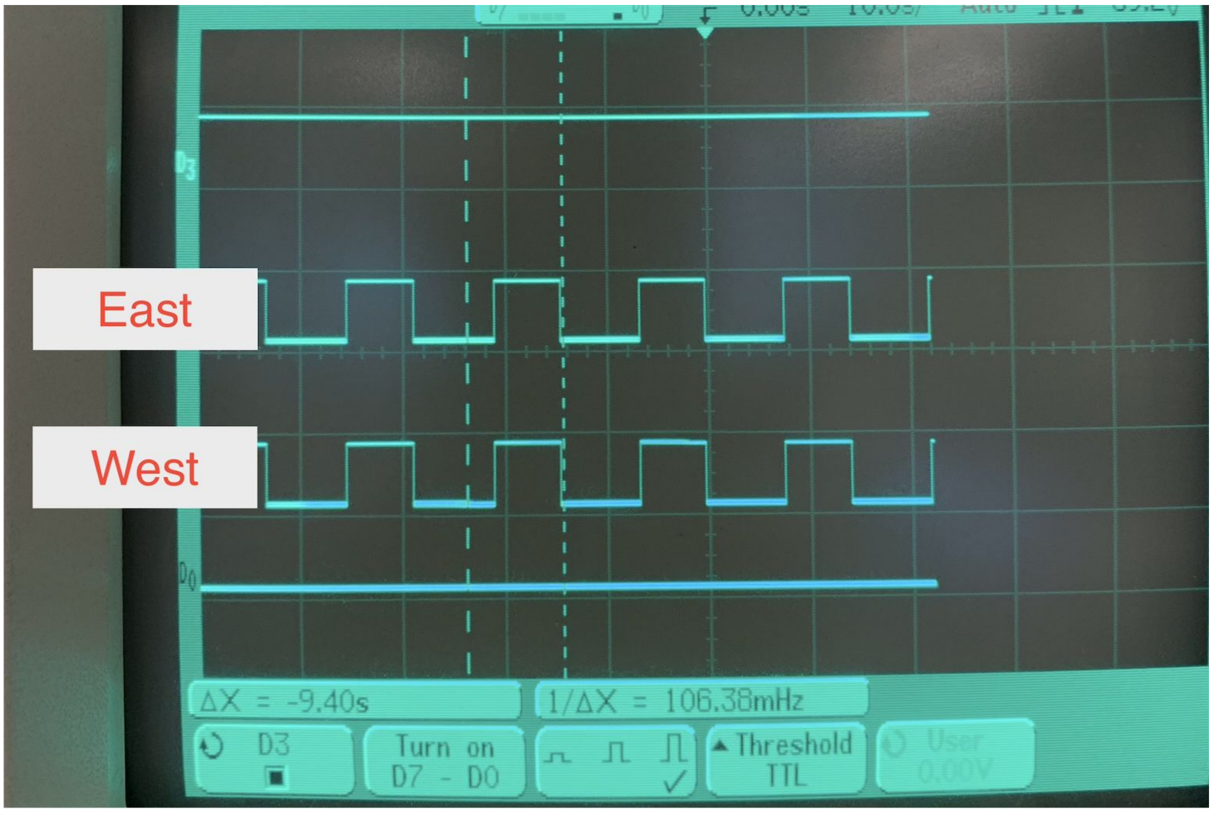


Figure 4. Screenshot of MSO showing the program running normally.

Here we can see a second MSO trace, shown below in Figure 5, where you can see the input “left” of the joystick as well as the lights for East Right and North and East. You can see the lights working as normal switching between EW and NS but then a press of the sensor, EReq (Joystick press “left”) is turned ON. As such, the lights for the rest are off and the light for East Right Turn is turned on for 3 time units. Once the turn is over, the other lights are turned on for a shorter period of time. (6 Time units instead of 10).

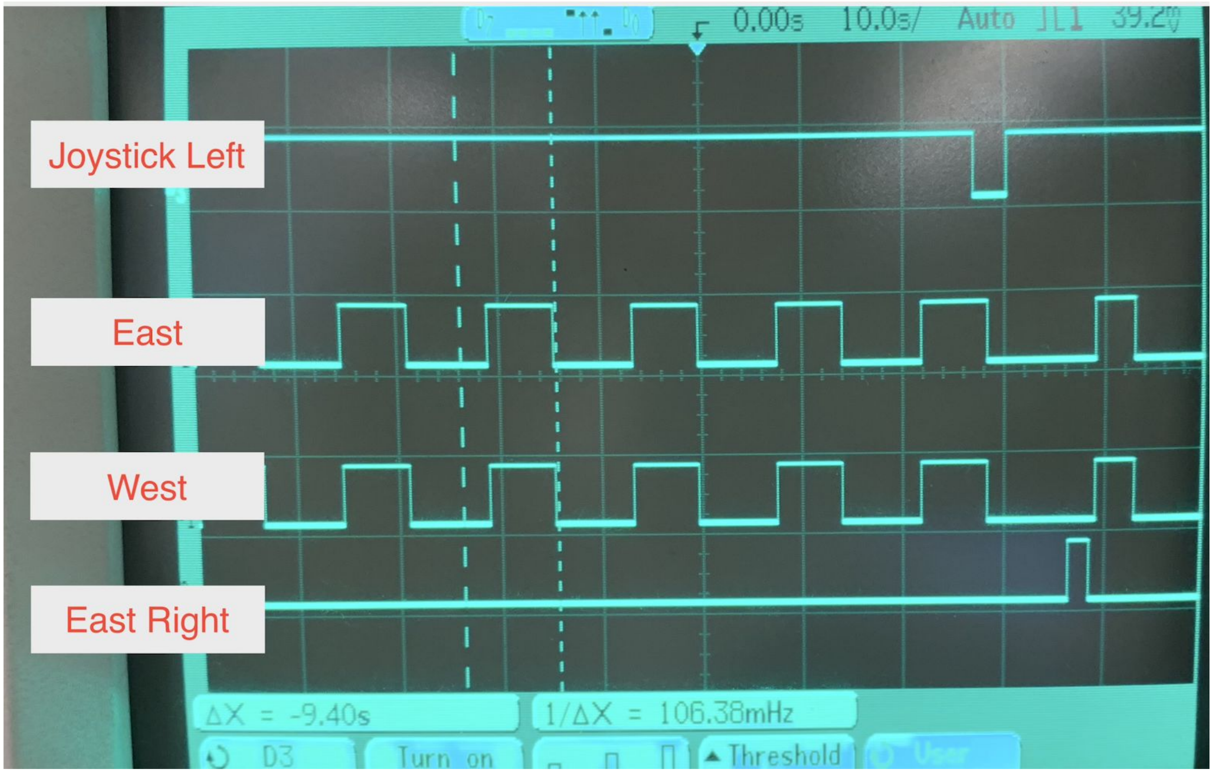


Figure 5. Screenshot of MSO

Conclusion:

References:

[1] Jidong Wang, “Lab 3 _ C Compiler and Traffic Lights .docx”, ed. Melbourne, Australia: RMIT University, 2019. [Online]. Available at: https://rmit.instructure.com/courses/49819/files/6674350/download?wrap=1

Appendix:

Shown below, is the code that was used for Lab 3:

```
#include <stm32f10x_cl.h>
#include "DspCode.h"
#define DELAY_C 0x0                    // this needs to be tried out to finalize
#define North 0x0100
#define NorthA 0x0200
#define South 0x0400
#define SouthA 0x0800
#define West 0x1000
```

```
#define WestA 0x2000
#define East 0x4000
#define EastA 0x8000
enum light_state
{ NS, EW };
enum light_state state;
int joystick;
int j;
void
SystemInit ()
{
}

// this function read the joystick and remember if the joysticks are ever selected.
// this function can also be used as a delay function. The length of the delay is determined by
// delay_count.  This function can be called for one time unit delay.
int
Read_joystick ()
{
  int stick;
  stick = joystick;
  stick |= ~GPIOD->IDR;                    //read joystick each loop
  stick &= 0xf000;                //relevant bits
  return stick;
}

void
Welcome ()
{
  GLCD_Init ();                   /* Initialize graphical LCD display */
  GLCD_Clear (White);                      /* Clear graphical LCD display */
  GLCD_SetTextColor (White);
  GLCD_SetBackColor (Black);
  GPIOE->ODR = 0xff00;                     //For timing
  GLCD_DisplayString (0, 0, "---Traffic Light---");
  GLCD_DisplayString (1, 0, "-----Simulator----- ");
  GLCD_DisplayString (3, 0, " Simulates ");
  GLCD_DisplayString (4, 0, " North-South and ");
  GLCD_DisplayString (5, 0, " East-West ");
  GLCD_DisplayString (6, 0, " Intersection. ");
  GPIOE->ODR = 0x0000;                     //timing
}

void
waitF (int nOfUnits)
```

```
{
  int countWait = nOfUnits;
  countWait *= DELAY_C;

  while (countWait != 0)
    {
      joystick = Read_joystick ();
      countWait -= 0x1;
    }
}

int
main (void)
{
  int WS, EN, SE, NW, joystick2, mainDelay;
  state = NS;
  RCC->APB2ENR |= 1 << 6; // Enable GPIOE clock
  RCC->APB2ENR |= 1 << 5; //Enable GPIOD clock
  GPIOE->CRH = 0x33333333;         //Configure the GPIO for LEDs
//GPIOD->CRH = 0x00000000; //Configure the GPIO for Joystick - Input Analog
  joystick = 0;                       // this is used to collect the joystick reading, It remembers all
right turn request
  // initialized to 0, ie no right turning vehicles at all sides.
  mainDelay = 1;
  WS = 0, EN = 0, SE = 0, NW = 0;
  Welcome ();
  for (;;)
    {
      switch (state)
           {
           case NS:

             // all lights to be Red for one time unit
             // use the Read_joystick function to read the joystick and also delay
// one time unit
             GPIOE->ODR = 0;    // all red
             waitF (1);
             mainDelay = 10;
             if (SE == 1 && NW == 1)
               {
                 //both right arrows are lit and straight
                 lights are off for 3 TU.GPIOE->ODR = NorthA
                       |SouthA;
                 GLCD_DisplayString (8, 0, "South-Right &
North-Right Green.");
```

```
        waitF (3);
        //all red 1 TU
        GPIOE->ODR = 0;
        waitF (1);
        //set mainDelay to 6 for NS traffic
        mainDelay = 6;
        SE = 0;
        NW = 0;
        joystick &= 0xA000;
      }
    else if (SE == 1 && NW == 0)
      {
        //straight and right arrow green.
        GPIOE->ODR = SouthA | South;
        GLCD_DisplayString (8, 0, "South-Right &
South-North Green.");
        waitF (3);
        //right arrow red, straight continues gree
        while right
            lane is cleared.GPIOE->ODR = South;
        waitF (1);
        //set mainDelay to 6 for NS traffic
        mainDelay = 6;
        SE = 0;
        joystick &= 0xB000;
      }
    else if (NW == 1 && SE == 0)
      {
        //straight and right arrow green.
        GPIOE->ODR = NorthA | North;
        GLCD_DisplayString (8, 0, "North-Right &
North-South Green.");
        waitF (3);
        //right arrow red, straight continues gree
        while right
            lane is cleared.GPIOE->ODR = North;
        waitF (1);
        //set mainDelay to 6 for NS traffic
        mainDelay = 6;
        NW = 0;
        joystick &= 0xE000;
      }

    // SN straight green for 10 time units
    GPIOE->ODR = North | South;      // Turn on LEDs
```

```
            on N and S GLCD_DisplayString (8, 0, "North & South
Green.     ");
            waitF (mainDelay);
            //we check if any cars are waiting on the
            West side wanting to go right.joystick2 = joystick;
            joystick2 &= 0x2000;
            if (joystick2 == 0x2000)
             {
              GLCD_DisplayString (9, 0, "West Turn
Requested      ");
              WS = 1;
             }
            //we check for cars on the East wanting to
            go right.joystick2 = joystick;
            joystick2 &= 0x8000;
            if (joystick2 == 0x8000)
             {
              GLCD_DisplayString (9, 0, "East Turn
Requested      ");
              EN = 1;
             }

            state = EW;              // next state is EW
            break;
          case EW:

            GPIOE->ODR = 0;    // all red
            waitF (1);
            mainDelay = 10;
            if (WS == 1 && EN == 1)
             {
              //both right arrows are lit and straight
              lights are off for 3 TU.GPIOE->ODR = WestA
                   |EastA;

              GLCD_DisplayString (8, 0, "East-Right &
West-Right Green.");
              waitF (3);
              //all red 1 TU
              GPIOE->ODR = 0;
              waitF (1);
              //set mainDelay to 6 for NS traffic
              mainDelay = 6;
              WS = 0;
              EN = 0;
```

```
        joystick &= 0x5000;
      }
    else if (WS == 1 && EN == 0)
      {
        //straight and right arrow green.
        GPIOE->ODR = WestA | West;
        GLCD_DisplayString (8, 0, "West-Right &
West-East Green. ");
        waitF (3);
        //right arrow red, straight continues gree
        while right
            lane is cleared.GPIOE->ODR = West;

        waitF (1);
        //set mainDelay to 6 for NS traffic
        mainDelay = 6;
        WS = 0;
        joystick &= 0xD000;
      }
    else if (EN == 1 && WS == 0)
      {
        //straight and right arrow green.
        GPIOE->ODR = EastA | East;
        GLCD_DisplayString (8, 0, "East-Right &
East-West Green. ");
        waitF (3);
        //right arrow red, straight continues gree
        while right
            lane is cleared.GPIOE->ODR = East;
        waitF (1);
        //set mainDelay to 6 for NS traffic
        mainDelay = 6;
        EN = 0;
        joystick &= 0x7000;
      }

    // EW straight green for 10 time units
    GPIOE->ODR = East | West;        // Turn on LEDs o
    E and W GLCD_DisplayString (8, 0, "East& West Green
");
    waitF (mainDelay);
    //we check if any cars are waiting on the
    South side wanting to go right.joystick2 = joystick;
    joystick2 &= 0x4000;
    if (joystick2 == 0x4000)
```

```
          {
            SE = 1;
            GLCD_DisplayString (9, 0, "South Turn
Requested.     ");
          }
         //we check for cars on the North wanting t
         go right.joystick2 = joystick;
         joystick2 &= 0x1000;
         if (joystick2 == 0x1000)
          {
            NW = 1;
            GLCD_DisplayString (9, 0, "North Turn
Requested.      ");
          }

         state = NS;              // next state is NS

         break;

        default:
         GPIOE->ODR = 0x0000;
         GLCD_DisplayString (8, 0, "There is a problem! ");
         waitF (5);
         state = NS;
         break;
        }
    }
}
```