```
// Shing Yan (sy228@cornell.edu)
// Raymond Siow (rs234@cornell.edu)
// Cornell University, Ithaca, NY 14853, USA
// ECE 476
// Final Project - Traffic control
// Section: Mondays 4:30pm-7:30pm
// Last Updated: April 27th, 2003, 6:16 AM.

#include <Mega32.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <delay.h>

// other constants
#define T0reload            256-250
#define maxkeys             16
#define NOBUTTON            99

// traffic light FSM states
#define B_GOES_A_WALK       1
#define A_DONT_WALK         2
#define B_YELLOW            3
#define B_RED               4
#define A_GREEN             5
#define B_WALK              6
#define A_GOES_B_WALK       7
#define B_DONT_WALK         8
#define A_YELLOW            9
#define A_RED               10
#define B_GREEN             11
#define A_WALK              12

// sensor states
#define NO_CAR              31
#define IS_CAR              32

// traffic condition constants
#define LO                  13
#define MI                  14
#define HI                  16

// traffic sensor condition
#define READY_TO_ADJ        17
#define RECENTLY_ADJ        18

// time constants used
#define TEN_MS              10
#define HUNDRED_MS          100
#define HALF_SEC            500
#define FIVE_MIN            300000
#define L_CYCLE_LO          60*2
#define L_CYCLE_NM          90*2    //Normal
#define L_CYCLE_HI          120*2
#define LT_A_FLASH          15*2
#define LT_B_FLASH          15*2

// timer variables
int half_sec;
int traf_t;
int peds_t;

// traffic light related variables
int lightState;
int lgt_t_a;
int lgt_t_b;
int lgt_t_a_dw;
```

```
int lgt_t_b_dw;
int lgt_t_a_reload;
int lgt_t_b_reload;

// traffic condition
int traf_cond_a;
int traf_cond_b;

// sensors related variables
int sensorState_a1;
int sensorState_a2;
int car_q_time_a1;
int car_q_time_a2;

int sensorState_b1;
int sensorState_b2;
int car_q_time_b1;
int car_q_time_b2;

int car_q_time_a;
int car_cnt_a;
int time_base_a;

int car_q_time_b;
int car_cnt_b;
int time_base_b;

// some other constants
int hi = L_CYCLE_HI >> 1;
int mi = L_CYCLE_NM >> 1;
int lo = L_CYCLE_LO >> 1;

void light_fsm(void);
void traffic_fsm(void);
void adjust_traf(void);
void pedestrian_fsm(void);
void initalize(void);
void main(void);

// timer0 overflow ISR
interrupt [TIM0_OVF] void timer0_overflow(void) {

    // reload to force 1 mSec overflow
    TCNT0 = T0reload;

    // Decrement the three times if they are not already zero
    if (half_sec > 0) half_sec--;
    if (traf_t > 0) --traf_t;
    if (peds_t > 0) --peds_t;
}

// When magnetic field -> 1
// No field -> 0

// light_fsm is a function that implements the state machine used
// to control switching of the traffic lights
void light_fsm(void) {

    half_sec = HALF_SEC;

    /**********************************

    PORT A - DIRECTION A

    .0 - RED
    .1 - YELLOW
    .2 - GREEN
```

```
    .3 - WALK
    .4 - DON'T WALK

    PORT B - DIRECTION B

    .0 - RED
    .1 - YELLOW
    .2 - GREEN
    .3 - WALK
    .4 - DON'T WALK

    **********************************/

    switch (lightState) {
        case B_GOES_A_WALK:
            // decrement b's counter
            if (lgt_t_b > 0) --lgt_t_b;
            // if counter reaches don't walk... switch to a's don't walk state
            if (lgt_t_b < lgt_t_b_dw) lightState = A_DONT_WALK;
            // set a's timer
            break;
        case A_DONT_WALK:
            // set walk off on a
            PORTA.3 = 1;
            // toggle don't walk light while decrementing b's counter
            PORTA.4 = ~PORTA.4;
            // if reaches 5 sec before the end of don't walk state... go to next state
            if (lgt_t_b > 0) --lgt_t_b;
            if ((lgt_t_b >> 1) < 5) lightState = B_YELLOW;
            break;
        case B_YELLOW:
            // set traffic light to yellow on b
            PORTB.2 = 1;
            PORTB.1 = 0;
            // 2 sec before going to state
            if (lgt_t_b > 0) --lgt_t_b;
            if ((lgt_t_b >> 1) < 3) lightState = B_RED;
            // set don't walk light on a if not already
            PORTA.4 = 0;
            break;
        case B_RED:
            // set traffic light to red on b
            PORTB.1 = 1;
            PORTB.0 = 0;
            // 1 sec before going to next state
            if (lgt_t_b > 0) --lgt_t_b;
            if ((lgt_t_b >> 1) < 2) lightState = A_GREEN;
            break;
        case A_GREEN:
            // set traffic light to green on a
            PORTA.0 = 1;
            PORTA.2 = 0;
            // 1 sec before going to next state
            if (lgt_t_b > 0) --lgt_t_b;
            if ((lgt_t_b >> 1) < 1) lightState = B_WALK;
            break;
        case B_WALK:
            // set don't walk off and set walk on on b
            PORTB.4 = 1;
            PORTB.3 = 0;
            // 1 sec befire going to next state
            if (lgt_t_b > 0) --lgt_t_b;
            lightState = A_GOES_B_WALK;
            break;
        case A_GOES_B_WALK:
            // decrement a's counter
            if (lgt_t_a > 0) --lgt_t_a;
```

```
            // if counter reaches don't walk... switch to b's don't walk state
            if (lgt_t_a < lgt_t_a_dw) lightState = B_DONT_WALK;
            // set a's timer
            break;
        case B_DONT_WALK:
            // set walk off on b
            PORTB.3 = 1;
            // toggle don't walk light while decrementing a's counter
            PORTB.4 = ~PORTB.4;
            // if reaches 5 sec before the end of don't walk state... go to next state
            if (lgt_t_a > 0) --lgt_t_a;
            if ((lgt_t_a >> 1) < 5) lightState = A_YELLOW;
            break;
        case A_YELLOW:
            // set traffic light to yellow on a
            PORTA.2 = 1;
            PORTA.1 = 0;
            // 2 sec before going to state
            if (lgt_t_a > 0) --lgt_t_a;
            if ((lgt_t_a >> 1) < 3) lightState = A_RED;
            // set don't walk light on b if not already
            PORTB.4 = 0;
            break;
        case A_RED:
            // set traffic light to red on a
            PORTA.1 = 1;
            PORTA.0 = 0;
            // 1 sec before going to next state
            if (lgt_t_a > 0) --lgt_t_a;
            if ((lgt_t_a >> 1) < 2) lightState = B_GREEN;
            break;
        case B_GREEN:
            // set traffic light to green on b
            PORTB.0 = 1;
            PORTB.2 = 0;
            // 1 sec before going to next state
            if (lgt_t_a > 0) --lgt_t_a;
            if ((lgt_t_a >> 1) < 1) lightState = A_WALK;
            break;
        case A_WALK:
            // set don't walk off and set walk on on a
            PORTA.4 = 1;
            PORTA.3 = 0;
            // 1 sec befire going to next state
            if (lgt_t_a > 0) --lgt_t_a;
            lightState = B_GOES_A_WALK;
            adjust_traf();
            break;
    }
}

// This task checks the road condition every 10ms and adjust the variables
// of the traffic lights if necessary.
void traffic_fsm(void) {

    /**********************************

    PORT A - DIRECTION A

    .6 - SENSOR IN A1
    .7 - SENSOR IN A2

    PORT B - DIRECTION B

    .6 - SENSOR IN B1
    .7 - SENSOR IN B2
```

```
    **********************************/

    //update the traffic condition every 10ms

    traf_t = TEN_MS;

    // state machine for traffic light a1
    switch (sensorState_a1) {
        case NO_CAR:
            PORTC.3 = 1;
            car_q_time_a1 = 0;
            if (PINA.6 == 0){
                sensorState_a1 = IS_CAR;
            }
            break;
        case IS_CAR:
            if (PINA.6 == 1) {
                sensorState_a1 = NO_CAR;
                car_cnt_a++;
            } else {
            // increment waiting time
                car_q_time_a1++;
            }
            break;
    }
    // if waiting time at the sensor > 1 sec, we assume that it's stopped
    // this is when the wait time is incremented
    if (car_q_time_a1 > 100) {
        car_q_time_a1-=100;
        car_q_time_a+=2;
        PORTC.3 = ~PORTC.3;
    }

    // state machine for traffic light a2
    switch (sensorState_a2) {
        case NO_CAR:
            PORTC.3 = 1;
            car_q_time_a2 = 0;
            if (PINA.7 == 0) {
                sensorState_a2 = IS_CAR;
            }
            break;
        case IS_CAR:
            if (PINA.7 == 1){
                sensorState_a2 = NO_CAR;
                car_cnt_a++;
            } else {
            // increment waiting time
                car_q_time_a2++;
            }
            break;
    }
    // if waiting time at the sensor > 1 sec, we assume that it's stopped
    // this is when the wait time is incremented
    if (car_q_time_a2 > 100) {
        car_q_time_a2-=100;
        car_q_time_a+=2;
        PORTC.3 = ~PORTC.3;
    }

    // state machine for traffic light b1
    switch (sensorState_b1) {
        case NO_CAR:
            PORTC.7 = 1;
            car_q_time_b1 = 0;
            if (PINB.6 == 0){
                sensorState_b1 = IS_CAR;
```

```
            }
            break;
        case IS_CAR:
            if (PINB.6 == 1) {
                sensorState_b1 = NO_CAR;
                car_cnt_b++;
            } else {
            // increment waiting time
                car_q_time_b1++;
            }
            break;
    }
    // if waiting time at the sensor > 1 sec, we assume that it's stopped
    // this is when the wait time is incremented
    if (car_q_time_b1 > 100) {
        car_q_time_b1-=100;
        car_q_time_b+=2;
        PORTC.7 = ~PORTC.7;
    }

    // state machine for traffic light a2
    switch (sensorState_b2) {
        case NO_CAR:
            PORTC.7 = 1;
            car_q_time_b2 = 0;
            if (PINB.7 == 0){
                sensorState_b2 = IS_CAR;
            }
            break;
        case IS_CAR:
            if (PINB.7 == 1){
                sensorState_b2 = NO_CAR;
                car_cnt_b++;
            } else {
            // increment waiting time
                car_q_time_b2++;
            }
            break;
    }
    // if waiting time at the sensor > 1 sec, we assume that it's stopped
    // this is when the wait time is incremented
    if (car_q_time_b2 > 100) {
        car_q_time_b2-=100;
        car_q_time_b+=2;
        PORTC.7 = ~PORTC.7;
    }

}

void adjust_traf(void) {

    int wait_a, rate_a, traf_a;
    int wait_b, rate_b, traf_b;
    int time_base;

    time_base = time_base_a + time_base_b;

    // set wait time for a1 and a2
    if (car_q_time_a > (time_base_b*3/4)) {
        wait_a = HI;
    } else if (car_q_time_a < (time_base_b/4)) {
        wait_a = LO;
    } else {
        wait_a = MI;
    }

    // set rate for a1 and a2
```

```
    if (((float)time_base_a)/car_cnt_a < 4) {
        rate_a = HI;
    } else if (((float)time_base_a*2)/car_cnt_a > 10) {
        rate_a = LO;
    } else {
        rate_a = MI;
    }

    traf_a = wait_a + rate_a;

    // determine the traffic in road a
    if (traf_a < MI*2) {
        traf_cond_a = LO;
        PORTC.0 = 0;
        PORTC.1 = 1;
        PORTC.2 = 1;
    } else if (traf_a < HI*2) {
        traf_cond_a = MI;
        PORTC.0 = 1;
        PORTC.1 = 0;
        PORTC.2 = 1;
    } else {
        traf_cond_a = HI;
        PORTC.0 = 1;
        PORTC.1 = 1;
        PORTC.2 = 0;
    }

    // set wait time for b1 and b2
    if (car_q_time_b > (time_base_a*3/4)) {
        wait_b = HI;
    } else if (car_q_time_b < (time_base_a/4)) {
        wait_b = LO;
    } else {
        wait_b = MI;
    }

    // set rate for b1 and b2
    if (((float)time_base_b)/car_cnt_b < 2) {
        rate_b = HI;
    } else if (((float)time_base_b)/car_cnt_b > 6) {
        rate_b = LO;
    } else {
        rate_b = MI;
    }

    traf_b = wait_b + rate_b;

    // determine the traffic in road b
    if (traf_b < MI*2) {
        traf_cond_b = LO;
        PORTC.4 = 0;
        PORTC.5 = 1;
        PORTC.6 = 1;
    } else if (traf_b < HI*2) {
        traf_cond_b = MI;
        PORTC.4 = 1;
        PORTC.5 = 0;
        PORTC.6 = 1;
    } else {
        traf_cond_b = HI;
        PORTC.4 = 1;
        PORTC.5 = 1;
        PORTC.6 = 0;
    }

    // reset variables for road a
```

```
    time_base_a = lgt_t_a_reload;
    car_q_time_a = 0;
    car_cnt_a = 0;

    // reset variables for road b
    time_base_b = lgt_t_b_reload;
    car_q_time_b = 0;
    car_cnt_b = 0;

    // now adjust the traffic light's changing time
    switch (traf_cond_a + traf_cond_b) {
        case (HI+HI):
            // traffic is HI on both sides
            lgt_t_a_reload = hi;
            lgt_t_b_reload = hi;
            break;
        case (HI+MI):
            // traffic is HI on one side, MI on the other
            lgt_t_a_reload = (traf_cond_a == MI)? hi - 20 : hi + 20;
            lgt_t_b_reload = (traf_cond_b == MI)? hi - 20 : hi + 20;
            break;
        case (HI+LO):
            // traffic is HI on one side, LO on the other
            lgt_t_a_reload = (traf_cond_a == MI)? mi - 20 : mi + 20;
            lgt_t_b_reload = (traf_cond_b == MI)? mi - 20 : mi + 20;
            break;
        case (MI+MI):
            // traffic is MI on both sides
            lgt_t_a_reload = mi;
            lgt_t_b_reload = mi;
            break;
        case (MI+LO):
            // traffic is MI on one side, LO on the other
            lgt_t_a_reload = (traf_cond_a == LO)? lo - 20 : lo + 20;
            lgt_t_b_reload = (traf_cond_b == LO)? lo - 20 : lo + 20;
            break;
        case (LO+LO):
            // traffic is MI on both sides
            lgt_t_a_reload = lo;
            lgt_t_b_reload = lo;
            break;
    }

    lgt_t_a = lgt_t_a_reload;
    lgt_t_b = lgt_t_b_reload;
}

// This task polls the pedestrian input and adjust the variables
// of the traffic lights if necessary.
void pedestrian_fsm(void) {

    /**********************************

    PORT A - DIRECTION A

    .5 - REQUEST BUTTON IN A

    PORT B - DIRECTION B

    .5 - REQUEST BUTTON IN B

    **********************************/

    peds_t = HUNDRED_MS;

    // check current light state and if the pedestrian button is pressed
    if ((lightState == A_GOES_B_WALK) && (PINA.5 == 0)) {
```

```
        // if traffic is low on the other road
        if (traf_cond_b < HI) {
            // if traffic on this road is not high
            if (traf_cond_a == LO) {
                peds_t = HALF_SEC*lgt_t_a_dw;
                time_base_a -= (lgt_t_a - lgt_t_a_dw);
                lgt_t_a = lgt_t_a_dw;
            } else if (traf_cond_a == MI) {
                peds_t = HALF_SEC*10;
                if (lgt_t_a - 20 > lgt_t_a_dw) {
                    time_base_a -= 20;
                    lgt_t_a = lgt_t_a - 20;
                } else {
                    time_base_a -= (lgt_t_a - lgt_t_a_dw);
                    lgt_t_a = lgt_t_a_dw;
                }
            }
            // otherwise deny request
        }
        // else denied
    } else if ((lightState == B_GOES_A_WALK) && (PINB.5 == 0)) {
        // if traffic is low on the other road
        if (traf_cond_a < HI) {
            // if traffic on this road is not high
            if (traf_cond_b == LO) {
                peds_t = HALF_SEC*lgt_t_a_dw;
                time_base_b -= (lgt_t_b - lgt_t_b_dw);
                lgt_t_b = lgt_t_b_dw;
            } else if (traf_cond_b == MI) {
                peds_t = HALF_SEC*10;
                if (lgt_t_b - 20 > lgt_t_b_dw) {
                    time_base_b -= 20;
                    lgt_t_b = lgt_t_b - 20;
                } else {
                    time_base_b -= (lgt_t_b - lgt_t_b_dw);
                    lgt_t_b = lgt_t_b_dw;
                }
            }
            // otherwise deny request
        }
        // else denied
    }
}

//================================
// initialize STK500 board
void initalize(void) {

    // init timer 0 to 1/uSec
    TCNT0 = T0reload;
    // turn on timer 0 overflow ISR
    TIMSK = 1;
    // prescalar to 64 (1 -> 1, 2 -> 8, 3 -> 64, 4 -> 256, 5 -> 1024
    TCCR0 = 3;

    // initialize variables for traffic light FSM
    half_sec = HALF_SEC;
    lgt_t_a_reload = L_CYCLE_NM >> 1;
    lgt_t_a = 0;
    lgt_t_b_reload = L_CYCLE_NM >> 1;
    lgt_t_b = lgt_t_b_reload;
    lgt_t_a_dw = LT_A_FLASH;
    lgt_t_b_dw = LT_B_FLASH;
    peds_t = HUNDRED_MS;
    traf_t = TEN_MS;
    car_cnt_a = 0;
    car_q_time_a = 0;
```

```
    sensorState_a1 = NO_CAR;
    sensorState_a2 = NO_CAR;
    sensorState_b1 = NO_CAR;
    sensorState_b2 = NO_CAR;
    time_base_b = lgt_t_b_reload;
    time_base_a = lgt_t_a_reload;

    // Init port A to show keyboard result (0-4) and accept inputs (5-7)
    DDRA.0 = 1; DDRA.1 = 1;
    DDRA.2 = 1; DDRA.3 = 1;
    DDRA.4 = 1; DDRA.5 = 0;
    DDRA.6 = 0; DDRA.7 = 0;
    PORTA.0 = 0;    PORTA.1 = 1;
    PORTA.2 = 1;    PORTA.3 = 0;
    PORTA.4 = 1;

    // Init port B to show keyboard result (0-4) and accept inputs (5-7)
    DDRB.0 = 1; DDRB.1 = 1;
    DDRB.2 = 1; DDRB.3 = 1;
    DDRB.4 = 1; DDRB.5 = 0;
    DDRB.6 = 0; DDRB.7 = 0;
    PORTB.0 = 1;    PORTB.1 = 1;
    PORTB.2 = 0;    PORTB.3 = 1;
    PORTB.4 = 0;

    // output indicator from sensor
    DDRC = 0xff;
    PORTC = 0xff;
    PORTC.1 = 0;
    PORTC.5 = 0;
    // input from pedestrian
    DDRD = 0x00;

    lightState = B_GOES_A_WALK;
    traf_cond_a = MI;
    traf_cond_b = MI;

    //enable sleep mode
    MCUCR = 0b01000000;
    #asm ("sei");

}

//================================
// set up the ports and timers
void main(void) {

    initalize();

    while(1) {
        if (half_sec == 0) light_fsm();
        if (traf_t == 0) traffic_fsm();
        if (peds_t == 0) pedestrian_fsm();

    }  // while
}  // main
```