```c
/*
 * GccApplication3.c
 *
 * Created: 2016-04-22 13:02:52
 *  Author: digpi03
 */

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

int race_active; // 0 ej aktivt, 1 aktivt
int passed_goal_line; // 0 ingen löpare har passerat mållinje, 1 en löpare har passerat
mållinje
int passed_left_goal_line; // 0 ej passerar, 1 passerat
int passed_right_goal_line; // 0 ej passerar, 1 passerat
volatile int count;
double measured_time = 0;


/************************************************************
 * SKÄRMEN
 *
 */
void setup_display() {

    // Function set
    DDRA = 0b11111111;
    _delay_ms(1);
    DDRD = 0b11100000;
    _delay_ms(1);
    highE();
    _delay_ms(1);
    PORTA = 0b00110000;
    _delay_ms(1);
    lowE();
    _delay_ms(1);

    //Display on
    display_command();
    PORTA = 0b00001100;
    _delay_ms(1);
    lowRS();
    _delay_ms(1);
    lowRW();
    _delay_ms(1);
    lowE();
}

// Clear display i databladet
void clear_display() {
    write_cmd(0b00000001); // Clear command
    _delay_ms(1); // behövs för att det ska fungera
}

void welcome_display() {
    write_string("PUSH THE BUTTON TO START.");
}
```

```
/*************************************************************
 * HJÄLPMETODER TILL SKÄRMEN
 *
 */

// Ska alltid användas innan kommando/tecken skickas
void display_command() {
    highE();
    highRW();
    highRS();
}

// Skriva kommando (ej tecken!)
void write_cmd(char c) {
    display_command();
    PORTA = c;
    _delay_ms(1);
    lowRS();
    _delay_ms(1);
    lowRW();
    _delay_ms(1);
    lowE();
}

// Skriva tecken till skärmen
void write_char(char ch) {
    display_command();
    PORTA = ch;
    _delay_ms(1);
    lowRW();
    _delay_ms(1);
    lowE();
}

// Skriva strängar till skärm
void write_string(char str[]) {
    int i = 0;
    while( str[i] != '\0') { // \0 null character in ASCII
        display_command();
        PORTA = str[i];
        _delay_ms(1);
        lowRW();
        _delay_ms(1);
        lowE();
        i++;
    }
}

// Höga och låga E, RS, R/W för displayen
void lowE() {
    PORTD &= 0b01111111;
}
void lowRW() {
    PORTD &= 0b10111111;
}
void lowRS() {
```

```
    PORTD &= 0b11011111;
}
void highE() {
    PORTD |= 0b10000000;
}
void highRW() {
    PORTD |= 0b01000000;
}
void highRS() {
    PORTD |= 0b00100000;
}



/*****************************************************************
 * DIODER
 *
 */
void turn_off_diods() {
    DDRB |= 0b00011000;
    PORTB &= 0b11100111;
}

void light_left_diod() {
    DDRB |= 0b00011000;
    _delay_ms(1);
    PORTB |= 0b00001000;
}
void light_right_diod() {
    DDRB |= 0b00011000;
    _delay_ms(1);
    PORTB |= 0b00010000;
}

/*****************************************************************
 * RACE
 *
 */
void setup_race() {
    race_active = 0;
}

void start_race() {
    clear_display();
    write_string("GET READY");

    _delay_ms(1000);
    clear_display();
    write_char('3');

    _delay_ms(1000);
    clear_display();
    write_char('2');

    _delay_ms(1000);
    clear_display();
    write_char('1');
```

```
    _delay_ms(1000);
    clear_display();
    write_string("GO");
}

void finish_race(char winner) {
    clear_display();
    write_string("WINNER: ");
    write_char(winner);
}

void race_logic(int track) { // track: 0 = höger bana, 1 = vänster bana
    if ( race_active == 1 ) {
        if (passed_goal_line == 0) {
            passed_goal_line = 1;
            start_clock();

            if (track == 0) { // Höger bana
                finish_race('2');
                light_right_diod();
                passed_right_goal_line = 1;
            }

            if (track == 1) { // Vänster bana
                finish_race('1');
                light_left_diod();
                passed_left_goal_line = 1;
            }

        } else if ( ( passed_left_goal_line == 0 && track == 1 ) || ( passed_right_goal_line
        == 0 && track == 0 ) ) {
            stop_clock();
            race_active = 0;
            _delay_ms(1);
            write_string("       DIFF: ");

            double time_in_seconds = measured_time/488.28125; // 488.28125 iom prescaler 8
            och 256, se längre ned

            int size = 7;
            char time_str[size]; // Tillåter max 7 tecken XXX.XXX float. Alltså max 999.999s
            tidsskillnad

            dtostrf(time_in_seconds, size, 3, time_str); //  Skapar char-array av double (3e
            parametern anger antal decimaler)
            write_string(time_str);
            write_string(" s");
        }
    }
}

/*************************************************************
 * INTERRUPT
 *
 */
void setup_interrupt() {

    MCUCSR = MCUCSR | 0b00001111; // Interrupt 0,1 (höger och vänster bana)
```

```
    MCUCSR = MCUCSR | 0b01000000; // Interrupt 2 (knappen)
    GICR = GICR | 0b11100000; // Sätt igång Interrupt 0,1,2

    sei(); // Enable global interrupt
}


// Höger bana
ISR(INT0_vect) {
    race_logic(0); // 0 = höger bana
}

// Vänster bana
ISR(INT1_vect) {
    race_logic(1); // 1 = vänster bana
}

// Knappen
ISR(INT2_vect) {

    if ( race_active == 0 ) {

        turn_off_diods();
        start_race();
        GIFR = 0b11100000; // Rensar Interrupt flags, vilket innebär att alla Interrupts som
        har körts då nedräkningen pågått ej räknas (motverkar tjuvstart)

        measured_time = 0;
        count = 0;
        race_active = 1;
        passed_right_goal_line = 0;
        passed_left_goal_line = 0;
        passed_goal_line = 0;
    }

}

// Overflow Interrupt (för klockan)
ISR(TIMER0_OVF_vect) {
    count++; // Räknas upp vid varje interrupt. I det här fallet 488.28125 per sekund
}


/*************************************************************
 * KLOCKAN
 *
 */
void clock_setup() {
    TCCR0 = TCCR0 | 0b00000010; // Prescaler 8
    TIMSK = TIMSK | 0b00000001; // Overflow interrupt aktivt
}
void start_clock() {
    count = 0;
}
void stop_clock() {
    measured_time = count;
}
```

```c
/*************************************************************
 * MAIN_METOD.
 * Programmets kärna
 *
 */
int main(void) {
    setup_display();
    setup_race();
    clear_display();
    welcome_display();
    setup_interrupt();
    clock_setup();
    turn_off_diods();

    while(1) {
    }
}
```