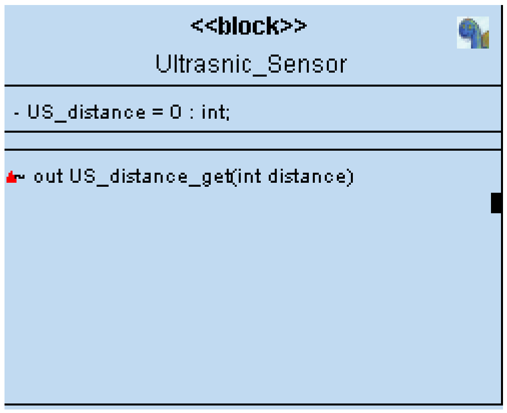
Modules level

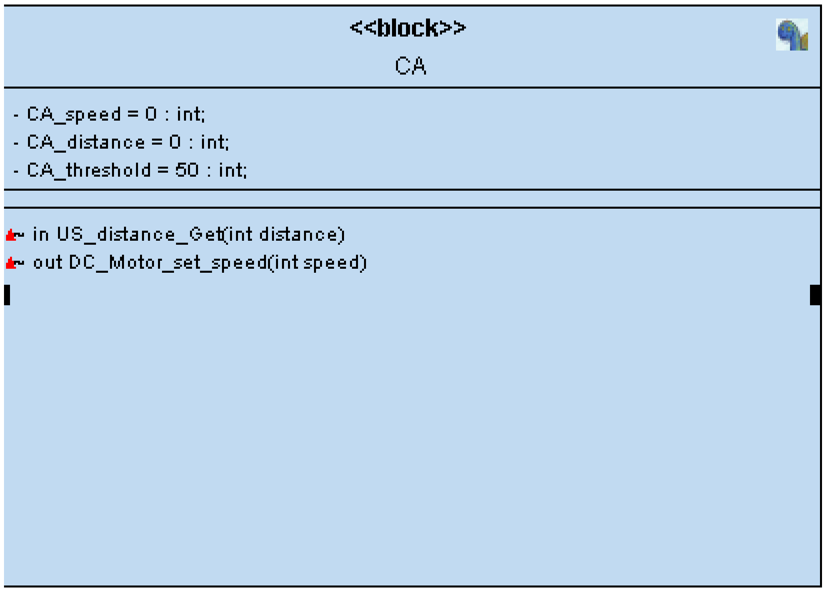
- **Ultrasonic Sensor module**
    - Attributes : US_distance
    - Signals : output signal to set the distance from sensor



- **collision Avoidance** :
    - Attributes : CA_distance , CA_speed , CA_threshold .
    - Signals :

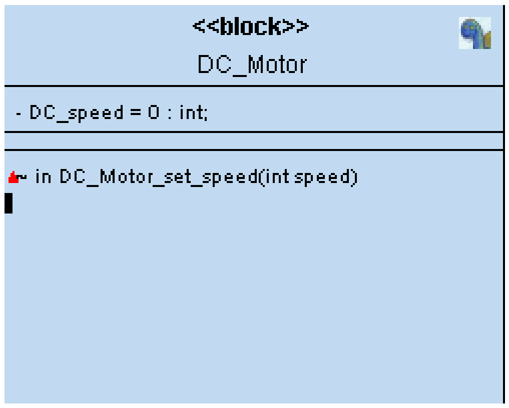
        * input signal to get the distance from sensor

        * output signal to set the speed of motor .



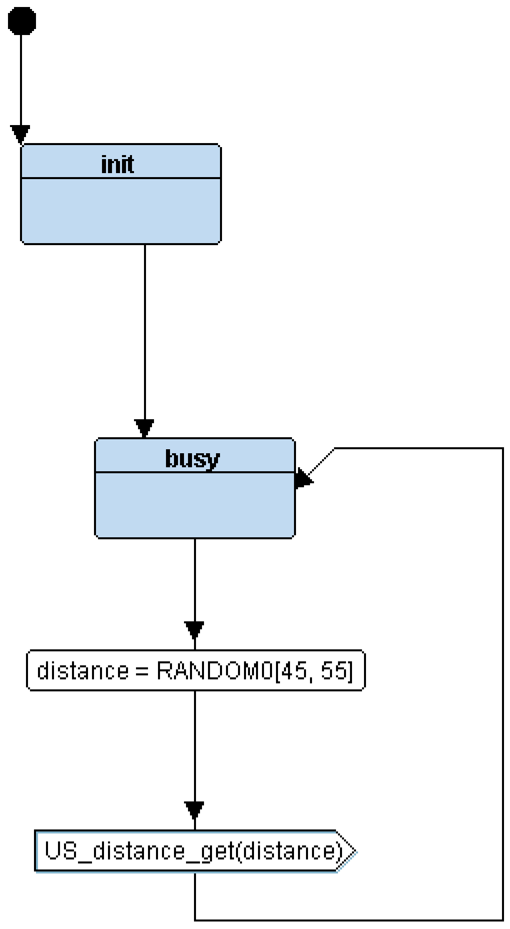
<<block>>
CA
- CA_speed = 0 : int;
- CA_distance = 0 : int;
- CA_threshold = 50 : int;
in US_distance_Get(int distance)
out DC_Motor_set_speed(int speed)

- **DC Motor :**
    - Attributes : DC_speed
    - Signals : input signal to set the speed of motor

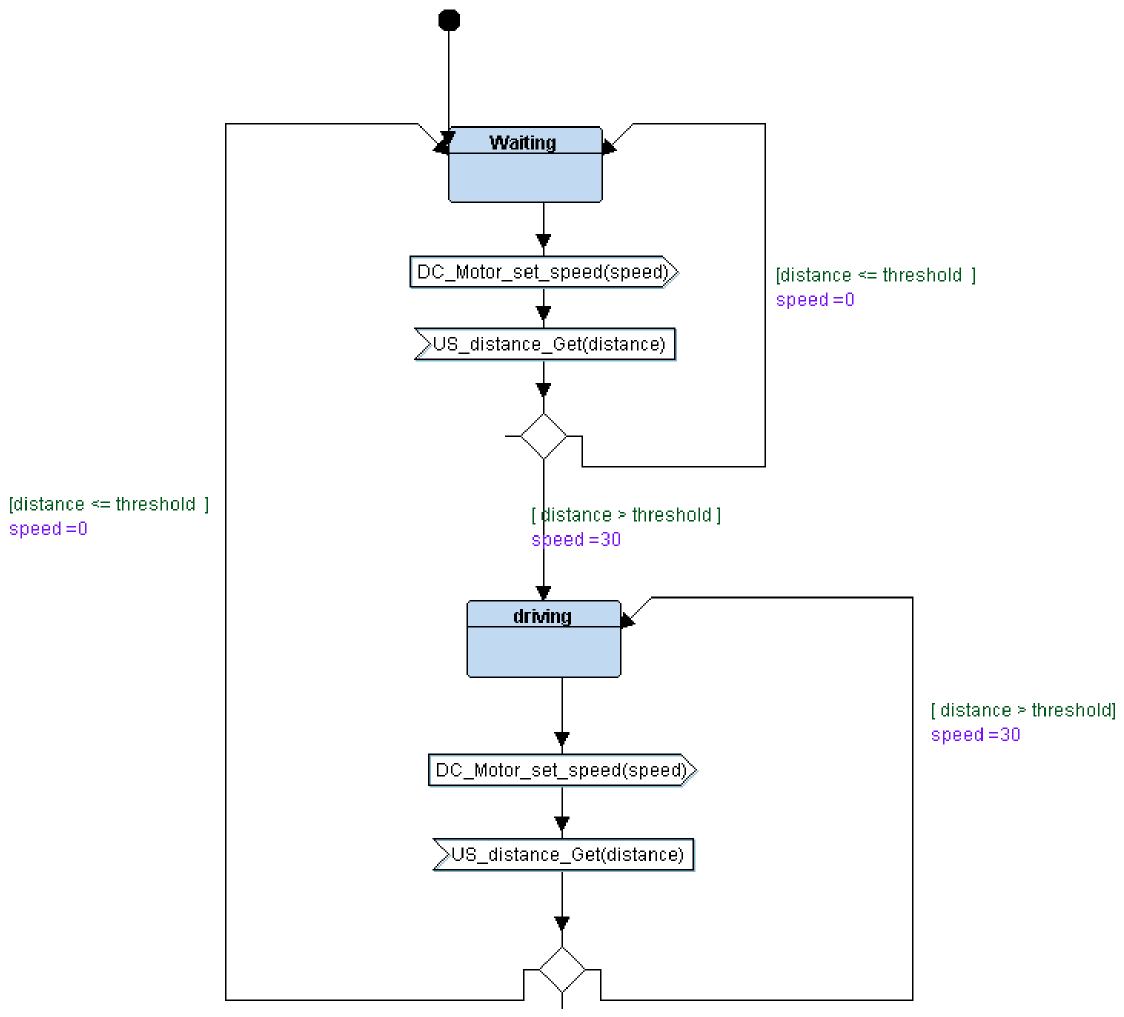


Logical design

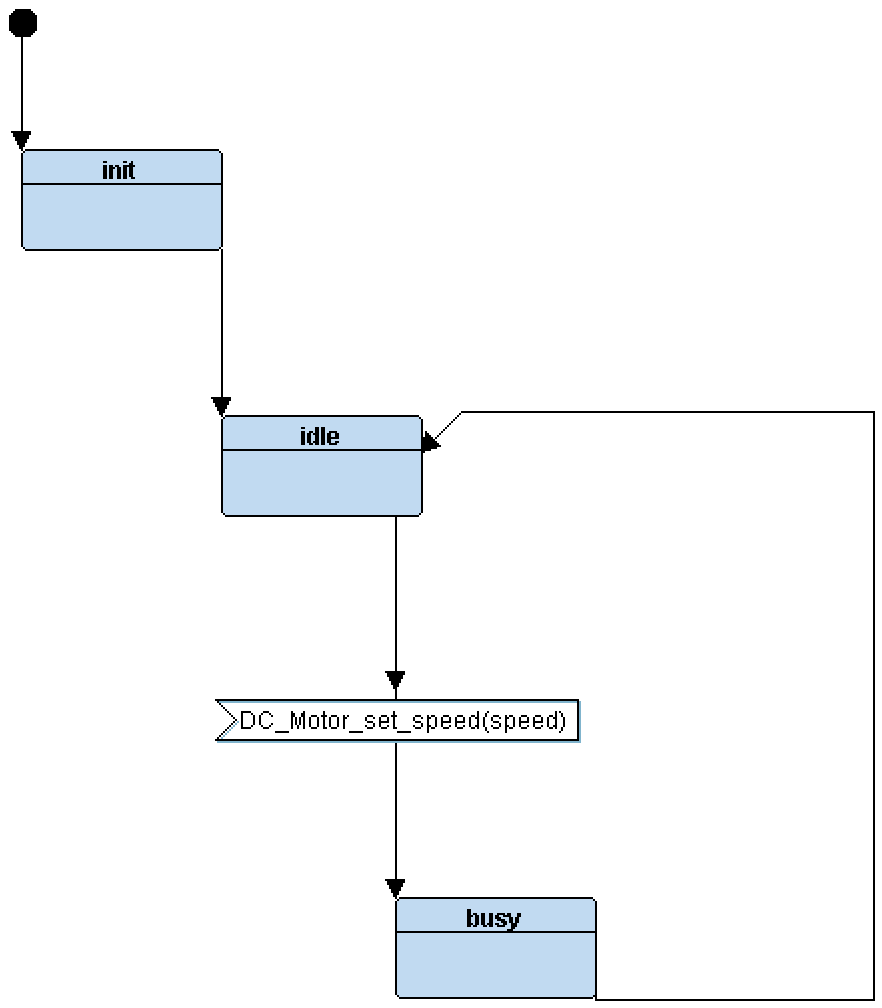
- **Ultrasonic Sensor modul**
    - at first initiate the ultrasonic sensor and it's drivers , the sensor enter busy state to calculate the distance and send it to the collision avoidance , and return to the busy state to get the distance again .



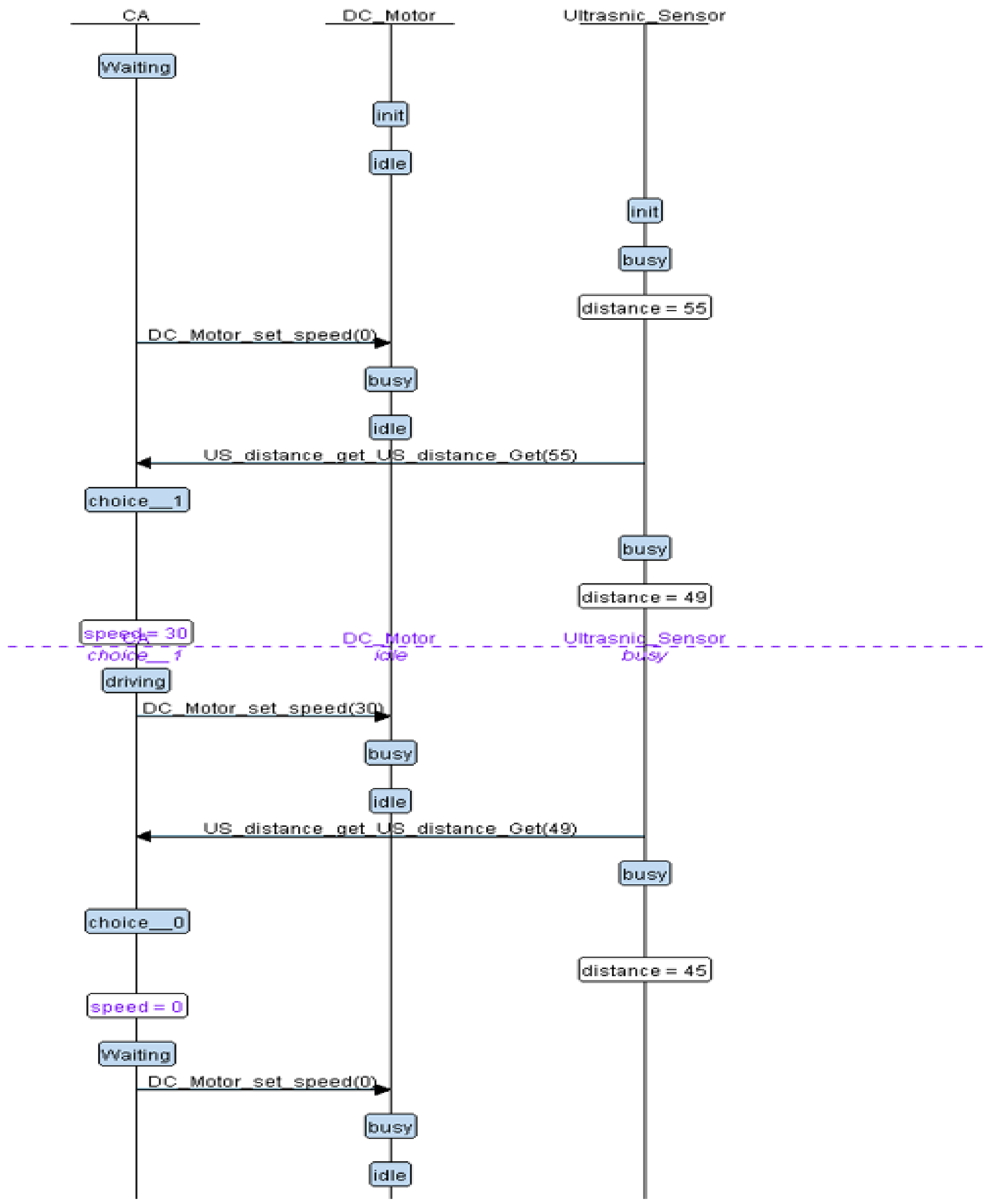
- **collision Avoidance :**
    - start in waiting state , set motor speed equal zero at first , get the distance from the sensor , if the distance less than the threshold return to waiting state and set speed = 0 , if not enter the driving mode and set speed =30 , and get the distance from the sensor again and do the same process on the new distance .



- **DC Motor :**
    - at first initiate the motor and it's drivers , the motor start at idle state , till get the speed from the collision avoidance and enter busy state to run motor with this speed , return to idle state after set speed till get a new speed .

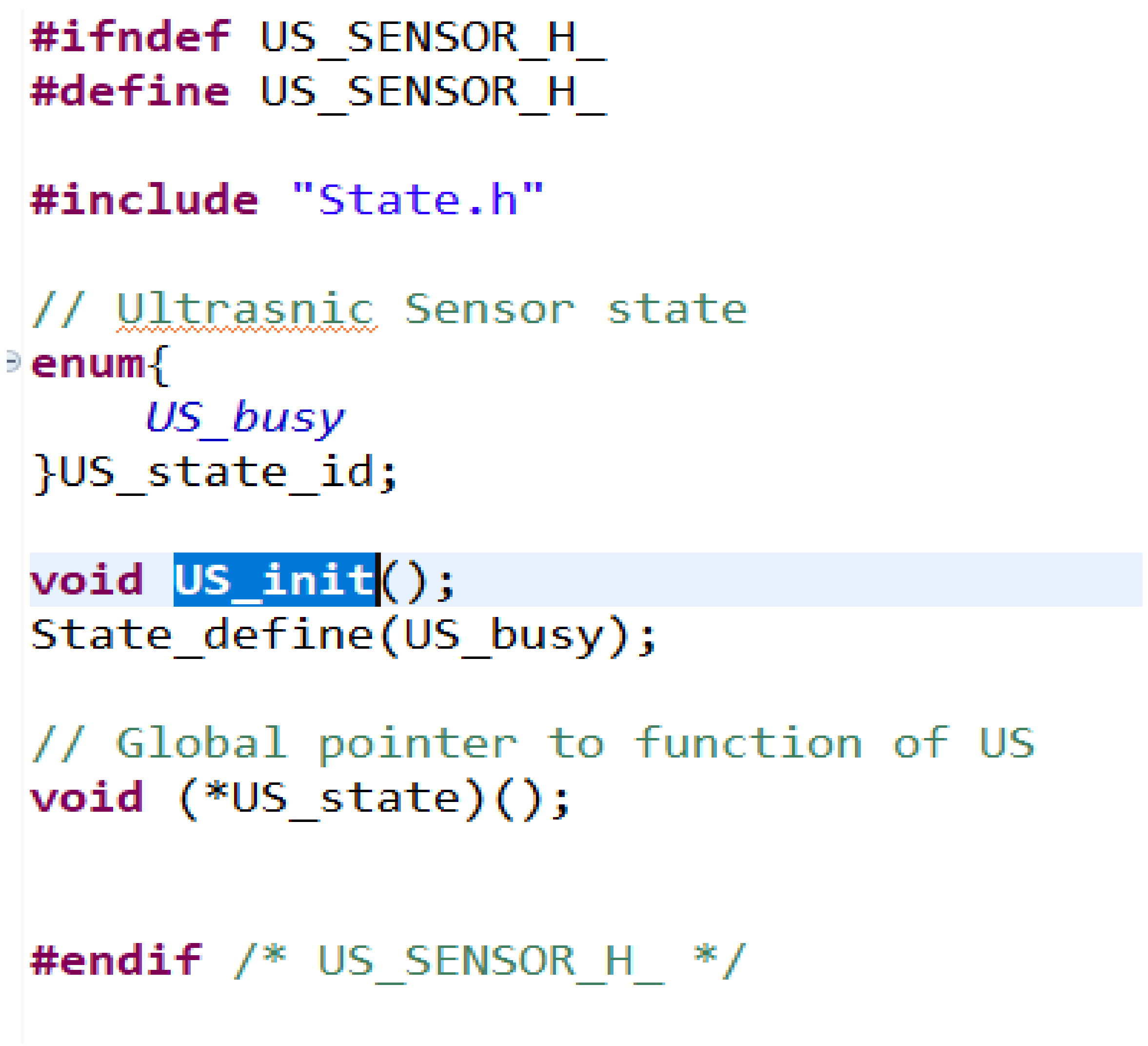


SW Logical verification

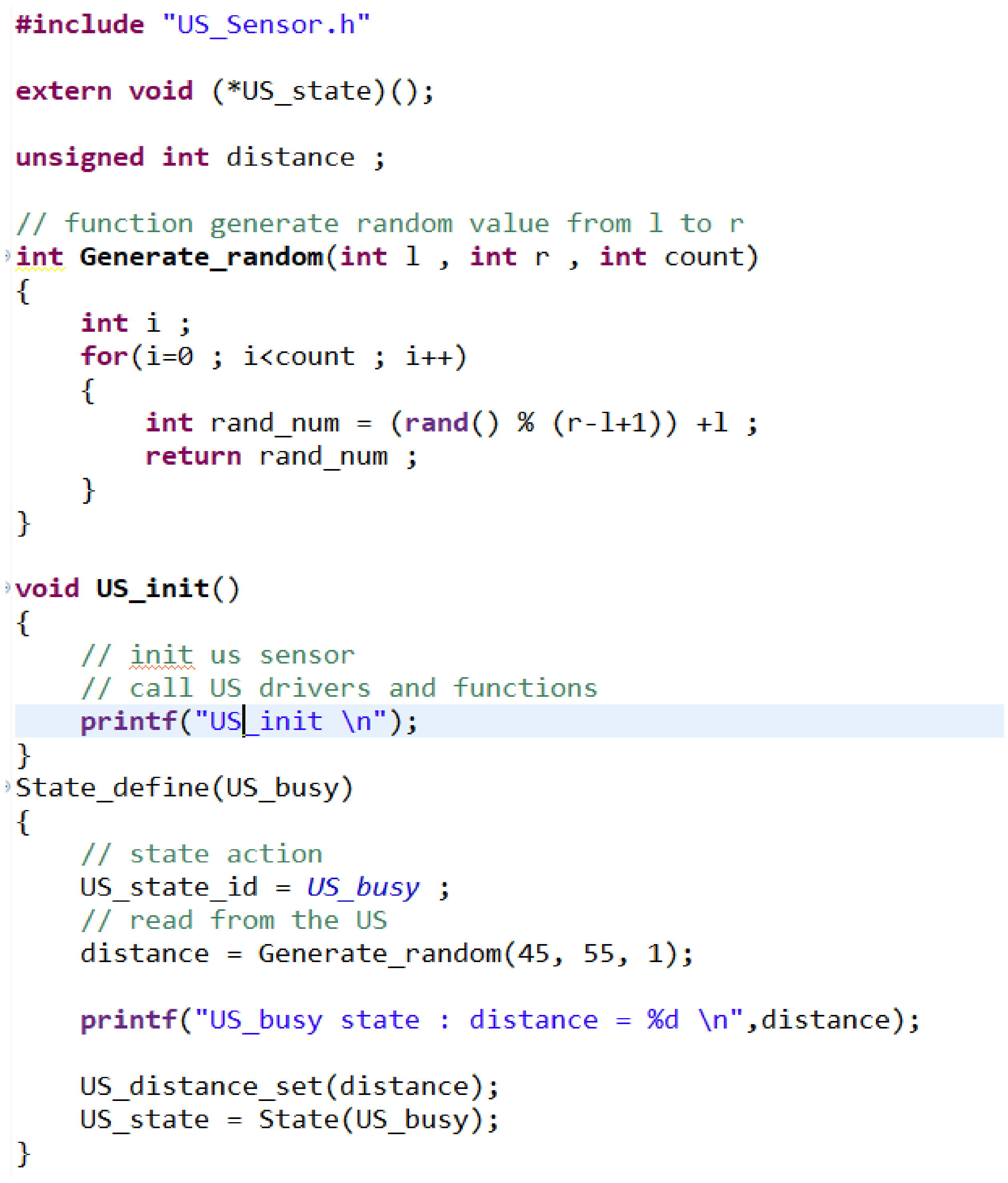


C implementation

- **Ultrasonic Sensor module**

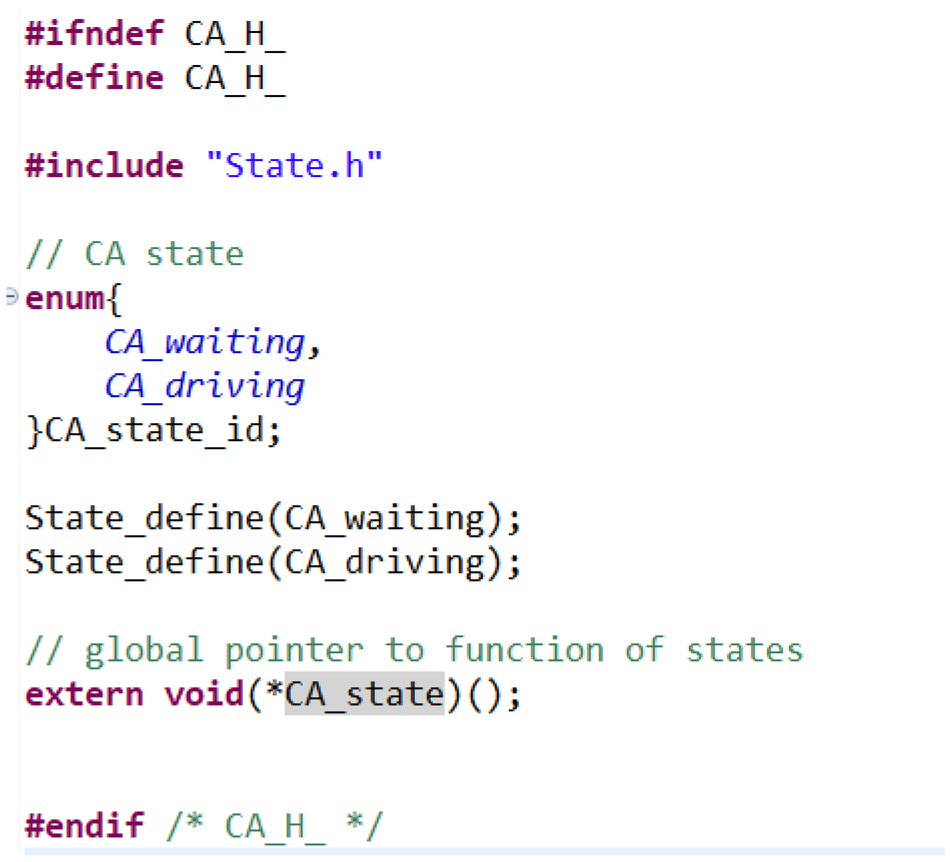


```c
#ifndef US_SENSOR_H_
#define US_SENSOR_H_

#include "State.h"

// Ultrasnic Sensor state
enum{
    US_busy
}US_state_id;

void US_init();
State_define(US_busy);

// Global pointer to function of US
void (*US_state)();


#endif /* US_SENSOR_H_ */
```

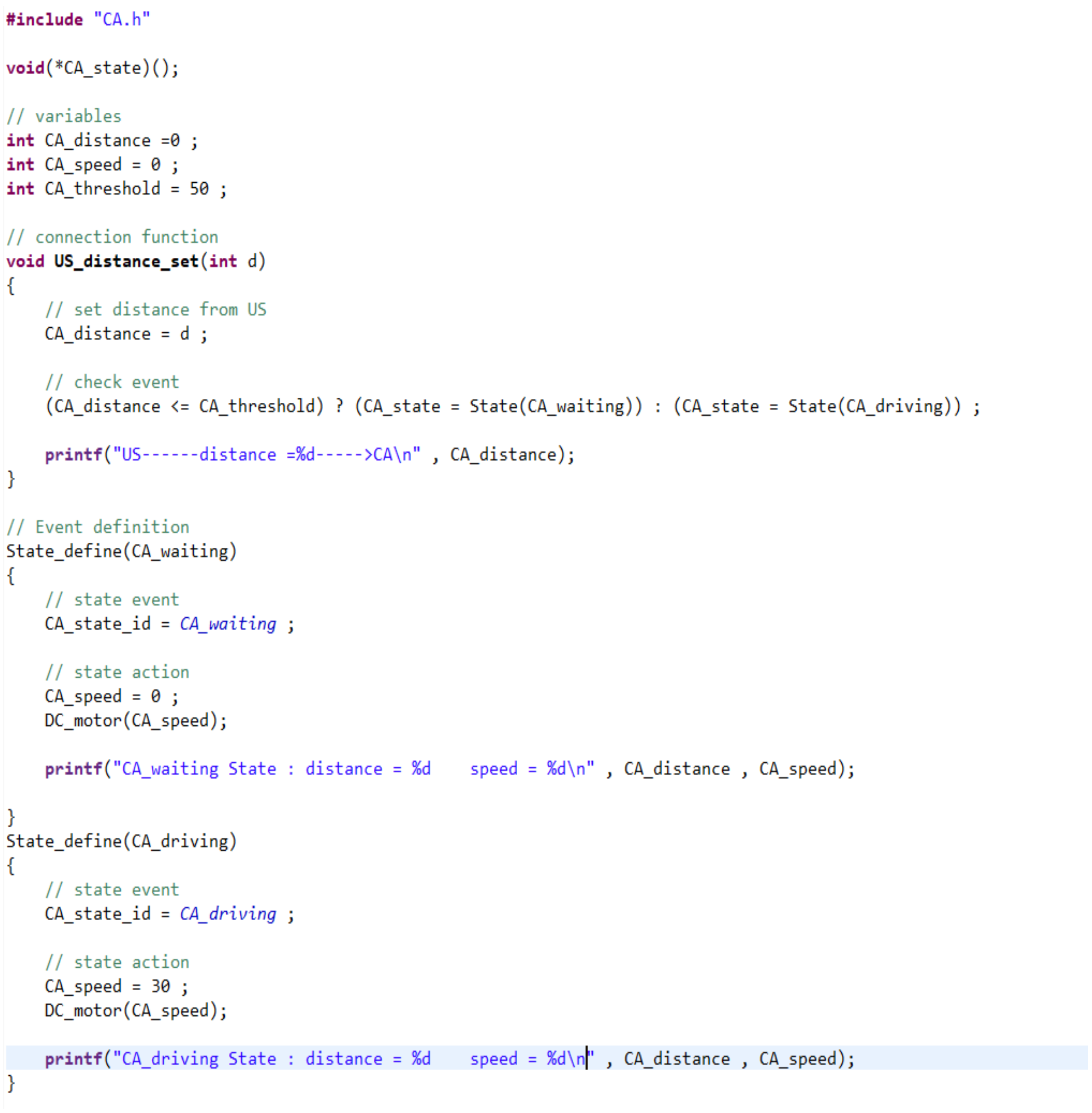


```c
#include "US_Sensor.h"

extern void (*US_state)();

unsigned int distance ;

// function generate random value from l to r
int Generate_random(int l , int r , int count)
{
    int i ;
    for(i=0 ; i<count ; i++)
    {
        int rand_num = (rand() % (r-l+1)) +l ;
        return rand_num ;
    }
}

void US_init()
{
    // init us sensor
    // call US drivers and functions
    printf("US_init \n");
}
State_define(US_busy)
{
    // state action
    US_state_id = US_busy ;
    // read from the US
    distance = Generate_random(45, 55, 1);

    printf("US_busy state : distance = %d \n",distance);

    US_distance_set(distance);
    US_state = State(US_busy);
}
```

- **collision Avoidance :**

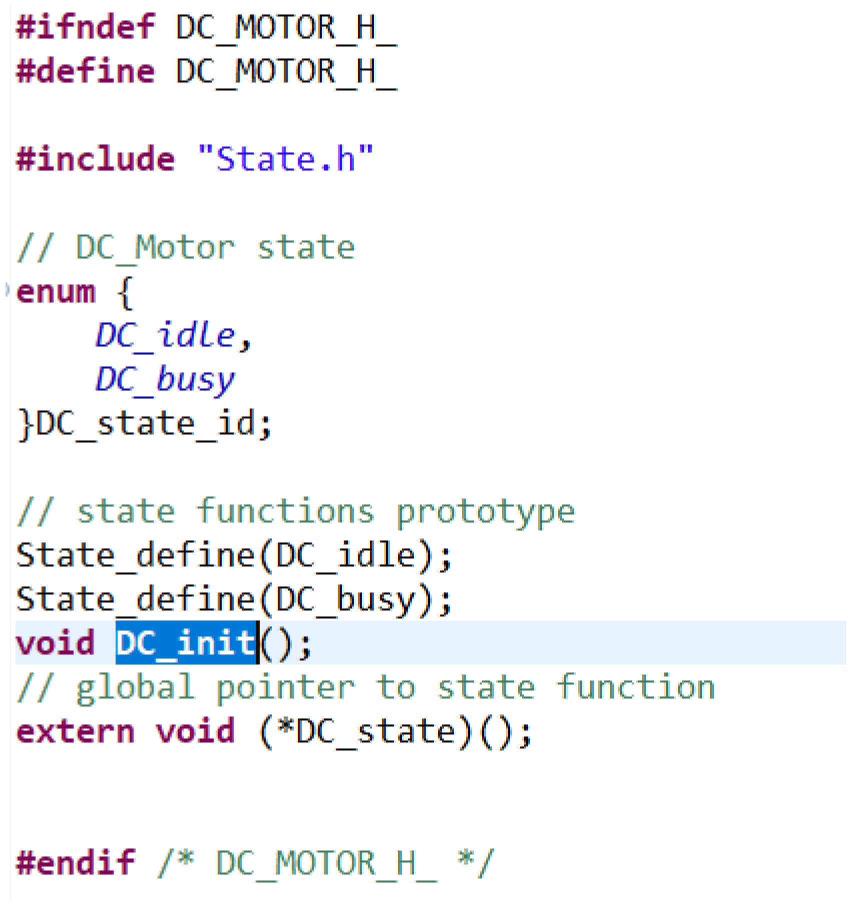


```c
#ifndef CA_H_
#define CA_H_

#include "State.h"

// CA state
enum{
	CA_waiting,
	CA_driving
}CA_state_id;

State_define(CA_waiting);
State_define(CA_driving);

// global pointer to function of states
extern void(*CA_state)();


#endif /* CA_H_ */
```

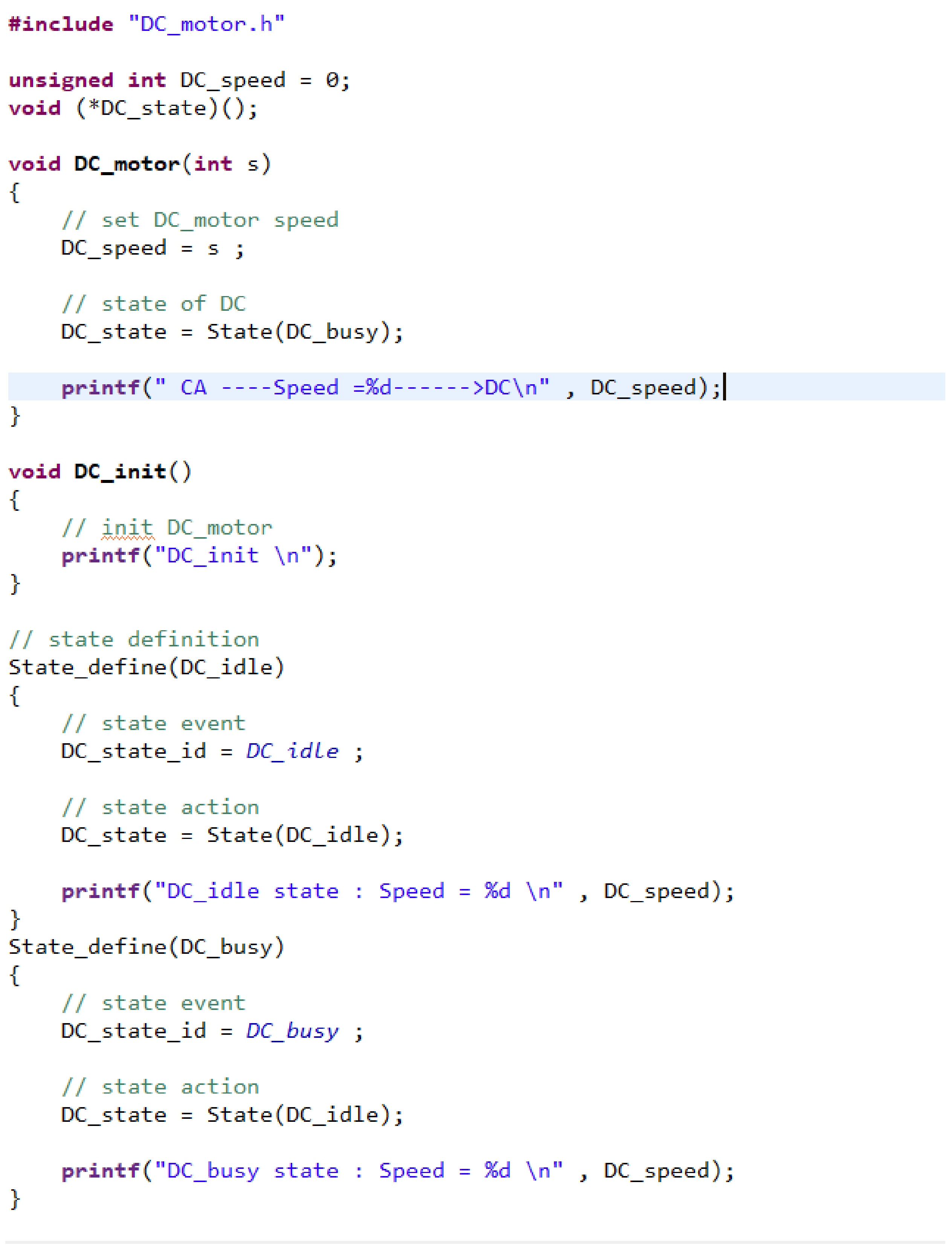


```c
#include "CA.h"

void(*CA_state)();

// variables
int CA_distance =0 ;
int CA_speed = 0 ;
int CA_threshold = 50 ;

// connection function
void US_distance_set(int d)
{
    // set distance from US
    CA_distance = d ;

    // check event
    (CA_distance <= CA_threshold) ? (CA_state = State(CA_waiting)) : (CA_state = State(CA_driving)) ;

    printf("US------distance =%d----->CA\n" , CA_distance);
}

// Event definition
State_define(CA_waiting)
{
    // state event
    CA_state_id = CA_waiting ;

    // state action
    CA_speed = 0 ;
    DC_motor(CA_speed);

    printf("CA_waiting State : distance = %d    speed = %d\n" , CA_distance , CA_speed);

}
State_define(CA_driving)
{
    // state event
    CA_state_id = CA_driving ;

    // state action
    CA_speed = 30 ;
    DC_motor(CA_speed);

    printf("CA_driving State : distance = %d    speed = %d\n" , CA_distance , CA_speed);
}
```

- **DC Motor** :



```c
#ifndef DC_MOTOR_H_
#define DC_MOTOR_H_

#include "State.h"

// DC_Motor state
enum {
    DC_idle,
    DC_busy
}DC_state_id;

// state functions prototype
State_define(DC_idle);
State_define(DC_busy);
void DC_init();
// global pointer to state function
extern void (*DC_state)();


#endif /* DC_MOTOR_H_ */
```

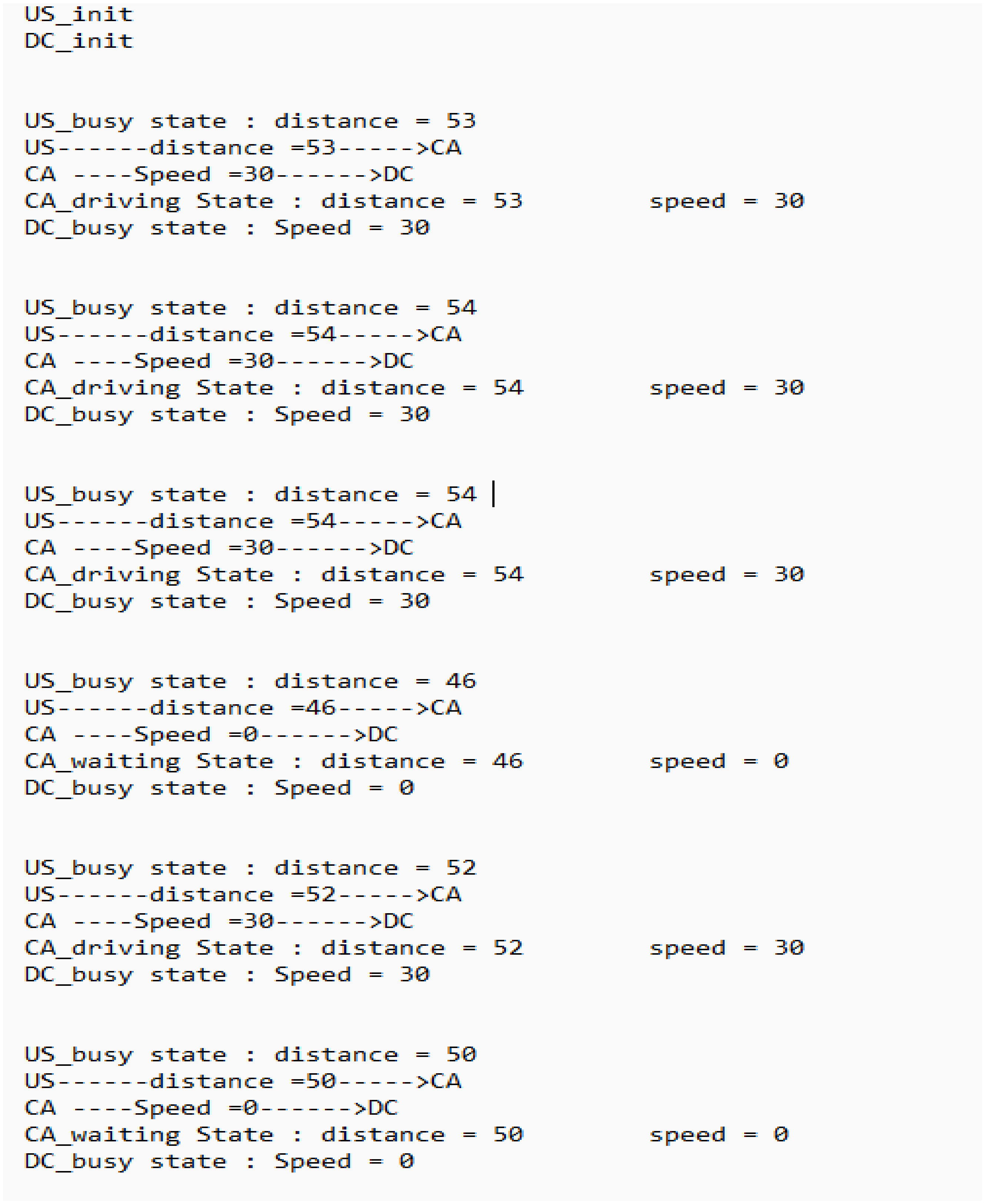


```c
#include "DC_motor.h"

unsigned int DC_speed = 0;
void (*DC_state)();

void DC_motor(int s)
{
    // set DC_motor speed
    DC_speed = s ;

    // state of DC
    DC_state = State(DC_busy);

    printf(" CA ----Speed =%d------>DC\n" , DC_speed);
}

void DC_init()
{
    // init DC_motor
    printf("DC_init \n");
}

// state definition
State_define(DC_idle)
{
    // state event
    DC_state_id = DC_idle ;

    // state action
    DC_state = State(DC_idle);

    printf("DC_idle state : Speed = %d \n" , DC_speed);
}
State_define(DC_busy)
{
    // state event
    DC_state_id = DC_busy ;

    // state action
    DC_state = State(DC_idle);

    printf("DC_busy state : Speed = %d \n" , DC_speed);
}
```

- **Main.c :**

```c
#include "CA.h"
#include "DC_motor.h"
#include "US_Sensor.h"

void setup()
{
    // init all drivers
    US_init();
    DC_init();

    // set pointer of states for each block
    CA_state = State(CA_waiting);
    US_state = State(US_busy);
    DC_state = State(DC_idle);
}

void main()
{
    volatile int i ;

    setup();

    while(1)
    {
        // Call state of each block
        US_state();
        CA_state();
        DC_state();
    }
    for(i =0 ; i<=1000 ; i++);
}
```

OUTPUT



```
US_init
DC_init


US_busy state : distance = 53
US------distance =53----->CA
CA ----Speed =30------>DC
CA_driving State : distance = 53        speed = 30
DC_busy state : Speed = 30


US_busy state : distance = 54
US------distance =54----->CA
CA ----Speed =30------>DC
CA_driving State : distance = 54        speed = 30
DC_busy state : Speed = 30


US_busy state : distance = 54
US------distance =54----->CA
CA ----Speed =30------>DC
CA_driving State : distance = 54        speed = 30
DC_busy state : Speed = 30


US_busy state : distance = 46
US------distance =46----->CA
CA ----Speed =0------>DC
CA_waiting State : distance = 46        speed = 0
DC_busy state : Speed = 0


US_busy state : distance = 52
US------distance =52----->CA
CA ----Speed =30------>DC
CA_driving State : distance = 52        speed = 30
DC_busy state : Speed = 30


US_busy state : distance = 50
US------distance =50----->CA
CA ----Speed =0------>DC
CA_waiting State : distance = 50        speed = 0
DC_busy state : Speed = 0
```