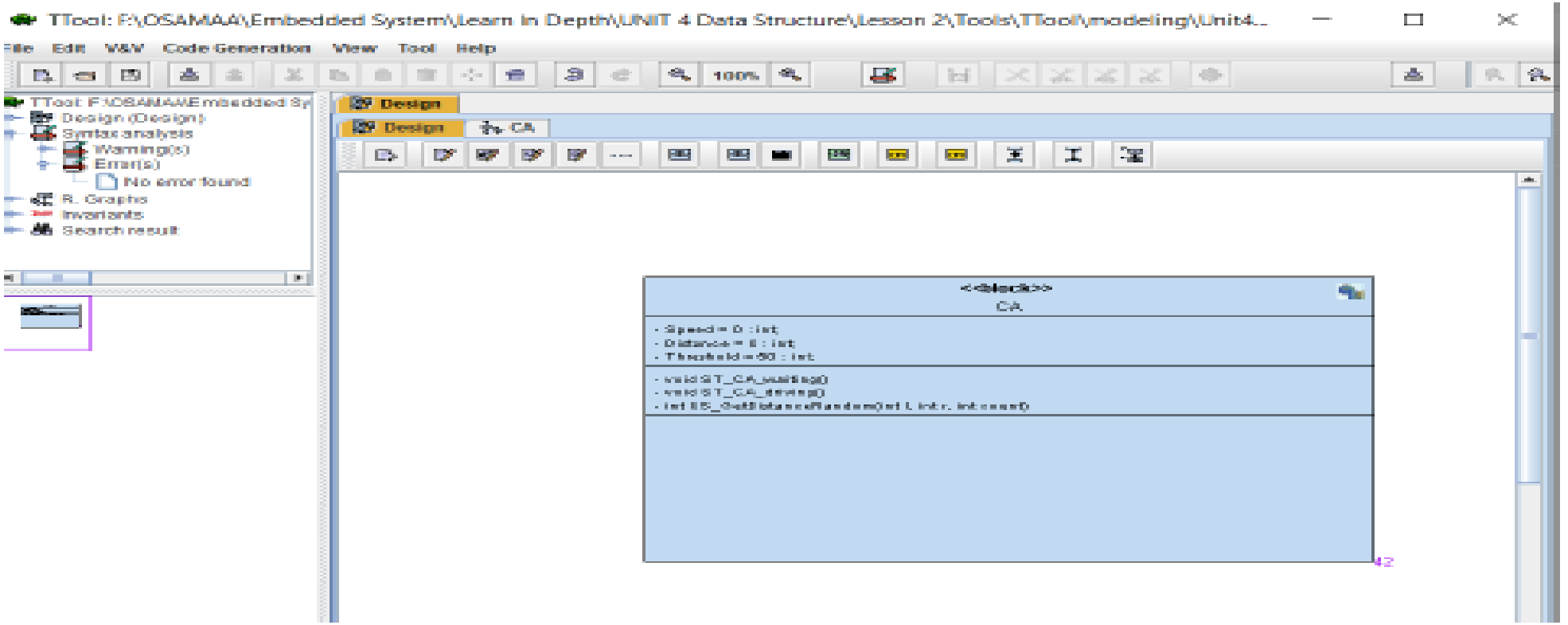
State Machine Simulation

State Machine

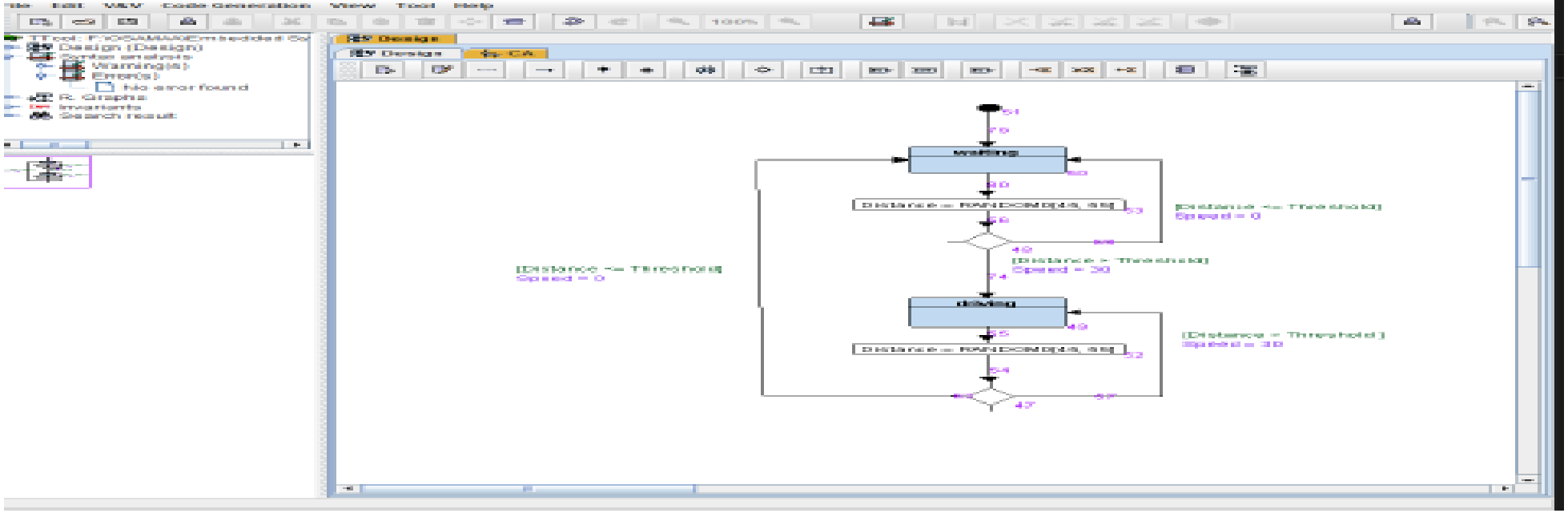
➢ Using 1Module

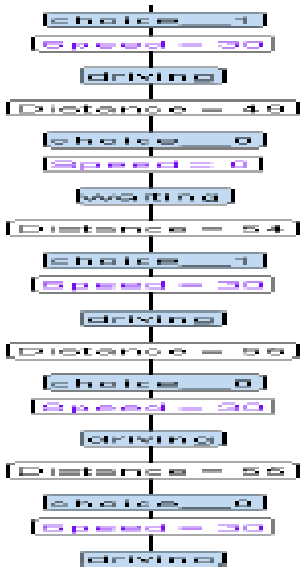
Simple Collision Avoidance to show how to implement using state diagram

There is a one block contain three functions.



The robot is starting from waiting state and switch to driving state if the Distance is greater than the value of threshold, and That shown in the figure below.


Design
waiting
driving
[Distance <= Threshold]
Speed = 0
[Distance > Threshold]
Speed = 30
distance = RANDOM(0, 35)


choice = 1
Speed = 30
driving
Distance = 48
choice = 0
Speed = 0
waiting
Distance = 54
choice = 1
Speed = 30
driving
Distance = 56
choice = 0
Speed = 30
driving
Distance = 56
choice = 0
Speed = 30
driving

USING C

Implementation Using C

```c
/*
 *   Topic   :                                          State Machine Single Block
 *   File    :                                          __DATA_TYPE_H__
 *   Author  :                                          abdelfattahzakariaelbadry@gmail.com
 */

/*
 *   Custom Built In Data Types Definiation For Global Using
 */

#ifndef __DATA_TYPE_H__
#define __DATA_TYPE_H__

typedef volatile unsigned char                          vuint8_t;
typedef volatile signed char                            vsint8_t;
typedef volatile unsigned short int                     vuint16_t;
typedef volatile signed short int                       vsint16_t;
typedef volatile unsigned long int                      vuint32_t;
typedef volatile signed long int                        vsint32_t;
typedef volatile unsigned long long int                 vuint64_t;
typedef volatile signed long long int                   vsint64_t;
typedef volatile float                                  vfloat32_t;
typedef volatile double                                 vdouble64_t;
typedef volatile long double                            vdouble96_t;

#define Element_Type                                    vuint32_t
#define True                                            1

#endif
```

Data Type.h

```c
/*
 *   Topic    :                                                State Machine Single Block
 *   File     :                                                __SATE_H__
 *   Author   :                                                abdelfattahzakariaelbadry@gmail.com
 */


/*
 *   State Header File Aims To Build Smart Function Prototypes For Serveral Useable Function Names
 */


#ifndef __SATE_H__
#define __SATE_H__


#define State_define(_StateFunc_)                                  void ST_##_StateFunc_()

/*Alias Name To Diserable Function Name Only To Be Able To Assigin It To A Ptr Func*/
#define State(_StateFunc_)                                         ST_##_StateFunc_

```

State.h

```c
/*
 *   Topic    :                                          State Machine Single Block
 *   File     :                                          __CA_H__
 *   Author   :                                          abdelfattahzakariaelbadry@gmail.com
*/

/*Header File For ca.c*/

#ifndef __CA_H__
#define __CA_H__


#include <stdio.h>
#include "datatype.h"
#include "state.h"


enum
{
    CA_Waiting ,
    CA_Driving
}CA_State_Id;


State_define(CA_Waiting);
State_define(CA_Driving);
int US_Generate_Random_Distance(vuint32_t beg , vuint32_t end);

/*A Pointer To A Function State_define With Ability To Be Callable In Other Files: void ST_##_StateFunc_() -->> ST_##_StateFunc: ST_0 , ST_1*/
extern void (*Ptr_CA_State) (void);
#endif

```

CA.H

```c
/*
 *   Topic    :                                                State Machine Single Block
 *   File     :                                                __CA_C__
 *   Author   :                                                abdelfattahzakariaelbadry@gmail.com
*/


/*ca.c: Implementation Of ca.h Prototype Functions*/


#include "ca.h"


/*Define Global Useable Variables: With Static Feature To Avoid Override Or Multi Definiation With Other Files*/
static vuint32_t CA_Spead= 0;
static vuint32_t CA_Distance= 0;
static vuint32_t CA_Threshold= 30;


/*A Pointer To A Function State_define With Ability To Be Callable In Other Files: void ST_##_StateFunc_() -->> ST_##_StateFunc: ST_0 , ST_1*/
/*Definiation: */
void (*Ptr_CA_State) (void);


int US_Generate_Random_Distance(vuint32_t beg , vuint32_t end)
{
    return ((rand() % (end - beg + 1)) + 1);
}


```

CA.C

```
42      /*b: Decision Making*/
43      (CA_Distance <= CA_Threshold)? (Ptr_CA_State= State(CA_Waiting)): (Ptr_CA_State= State(CA_Driving));
44
45      printf("Waiting State: distance= %u \tspeed= %u\n" , CA_Distance , CA_Spead);
46
47      return;
48  }
49  State_define(CA_Driving)
50  {
51      /*Assign State Id OR Name To Enum Object: */
52      CA_State_Id= CA_Driving;
53
54      /*State Action: Disable Motor Enable Signal*/
55      CA_Spead= 45;
56
57      /*State Check To The Decision Making Condition*/
58      /*a: Read Sensors Reads: Distance*/
59      CA_Distance= US_Generate_Random_Distance(45 , 90);
60      /*b: Decision Making*/
61      (CA_Distance <= CA_Threshold)? (Ptr_CA_State= State(CA_Waiting)): (Ptr_CA_State= State(CA_Driving));
62
63      printf("Driving State: distance= %u \tspeed= %u\n" , CA_Distance , CA_Spead);
64
```

CA.C Cont...

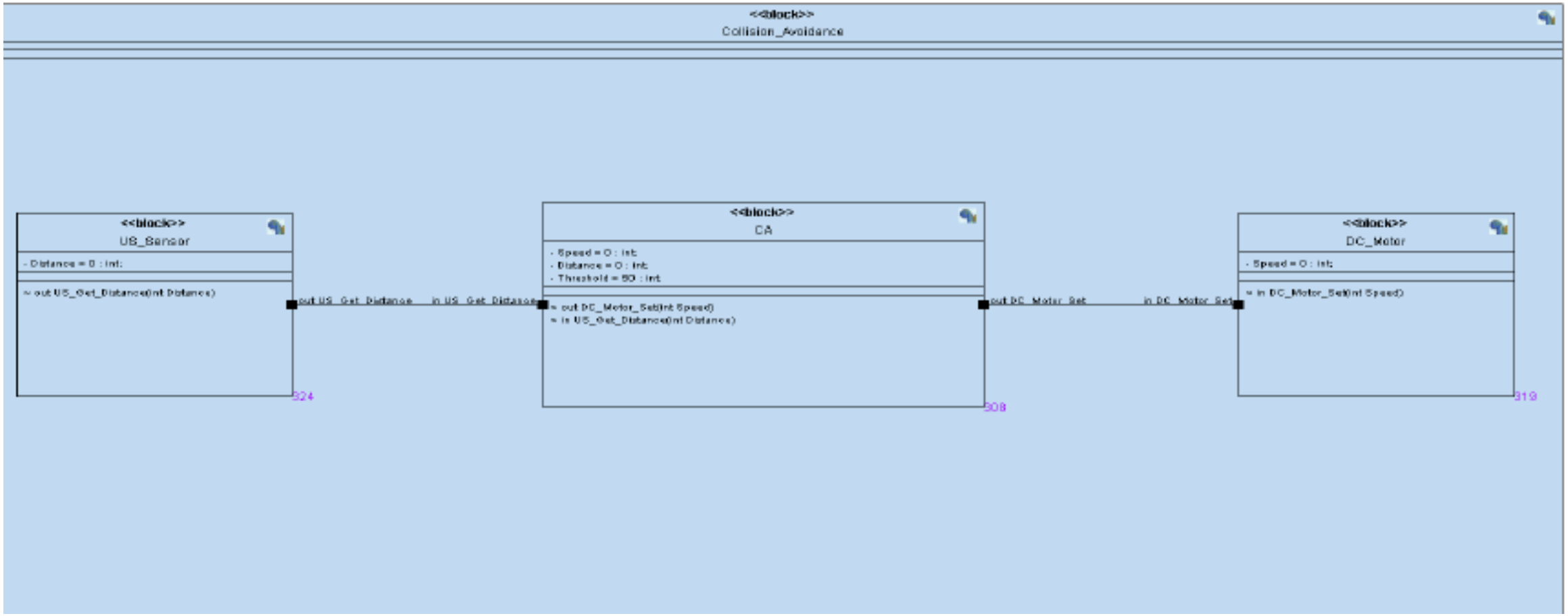
```c
/*
 *   Topic    :                                                State Machine Single Block
 *   File     :                                                __MAIN_C__
 *   Author   :                                                abdelfattahzakariaelbadry@gmail.com
 */
#include "ca.h"
void setup(void)
{
    /*Init All Drivers*/
    /*Init IRQ*/
    /*Init Hal Drivers*/
    /*Init Block*/
    /*Set State Pointer For Each Block: Initial State OR Startup|poweron State*/
    Ptr_CA_State= State(CA_Waiting);
    return;
}
int main(void)
{
    /*Init Essential Components*/
    setup();
    /*Start Infinte Runing*/
    vuint32_t i;
    while(True)
    {
        Ptr_CA_State();
        for(i= 0; i <= 15000; i++);
        Ptr_CA_State();
    }
    return 0;
}
```

Main.c

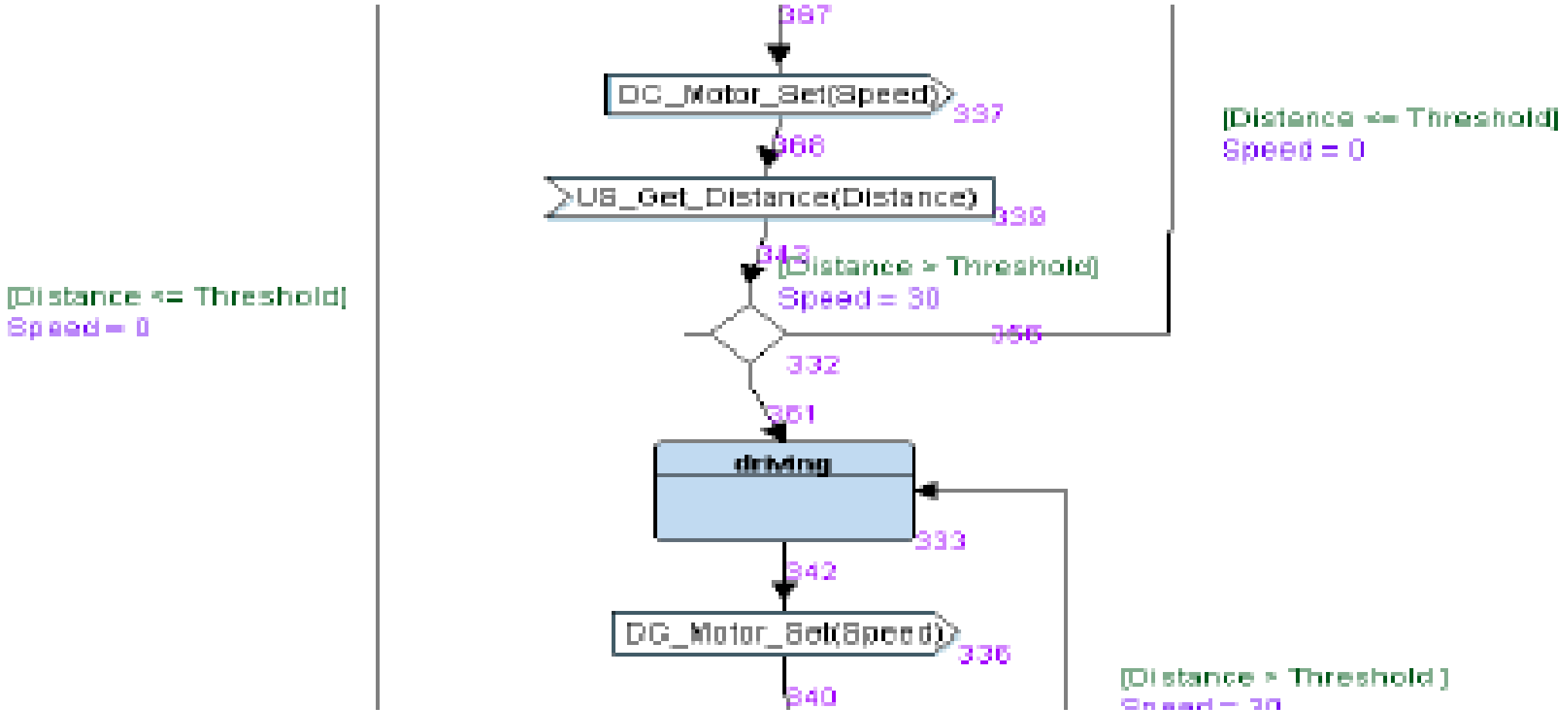
```
Driving State: distance= 42     speed= 45
Driving State: distance= 4      speed= 45
Waiting State: distance= 6      speed= 0
Waiting State: distance= 29     speed= 0
Waiting State: distance= 6      speed= 0
Waiting State: distance= 25     speed= 0
Waiting State: distance= 26     speed= 0
Waiting State: distance= 8      speed= 0
Waiting State: distance= 15     speed= 0
Waiting State: distance= 11     speed= 0
Waiting State: distance= 25     speed= 0
Waiting State: distance= 28     speed= 0
Waiting State: distance= 40     speed= 0
Driving State: distance= 6      speed= 45
Waiting State: distance= 33     speed= 0
Driving State: distance= 39     speed= 45
Driving State: distance= 9      speed= 45
Waiting State: distance= 36     speed= 0
Driving State: distance= 10     speed= 45
Waiting State: distance= 9      speed= 0
Waiting State: distance= 46     speed= 0
Driving State: distance= 10     speed= 45
```

Simulation Results

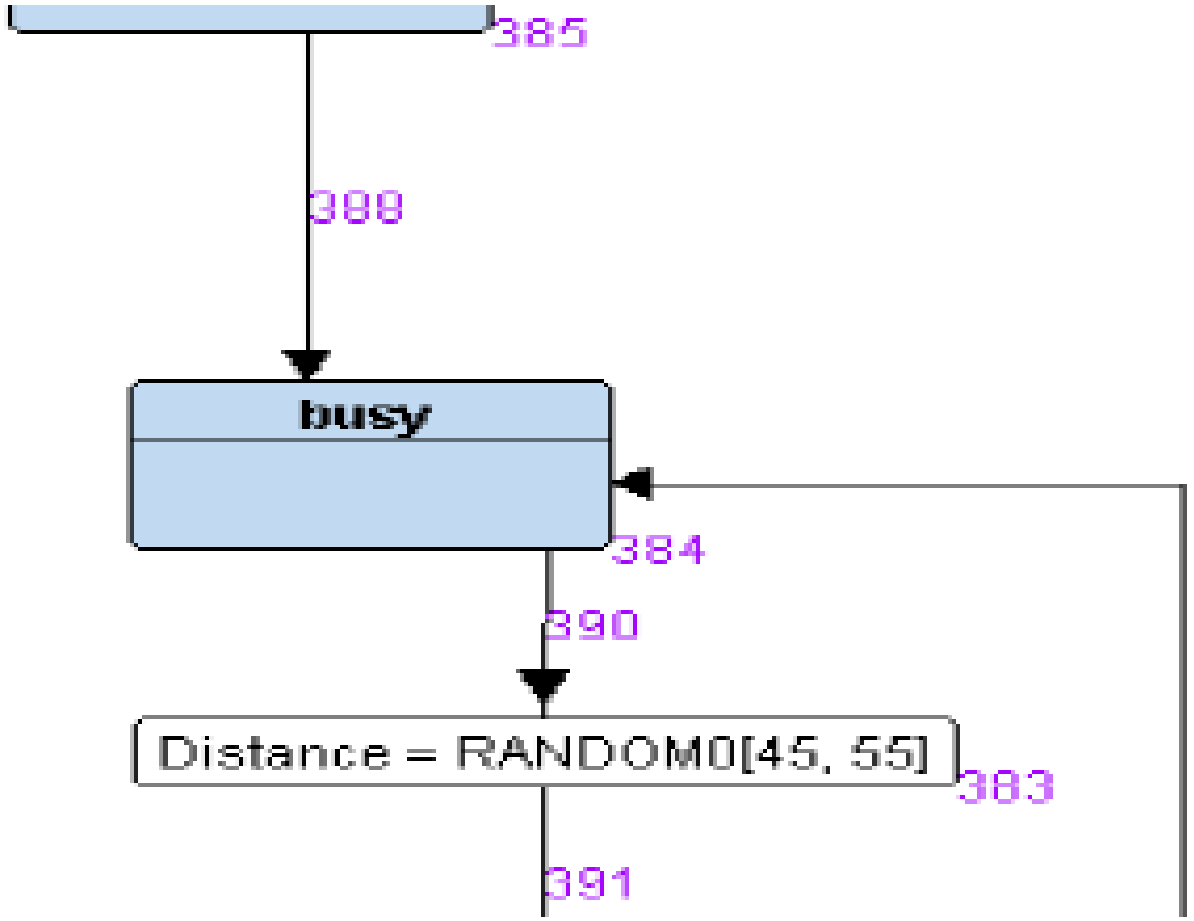
Using Multi Blocks



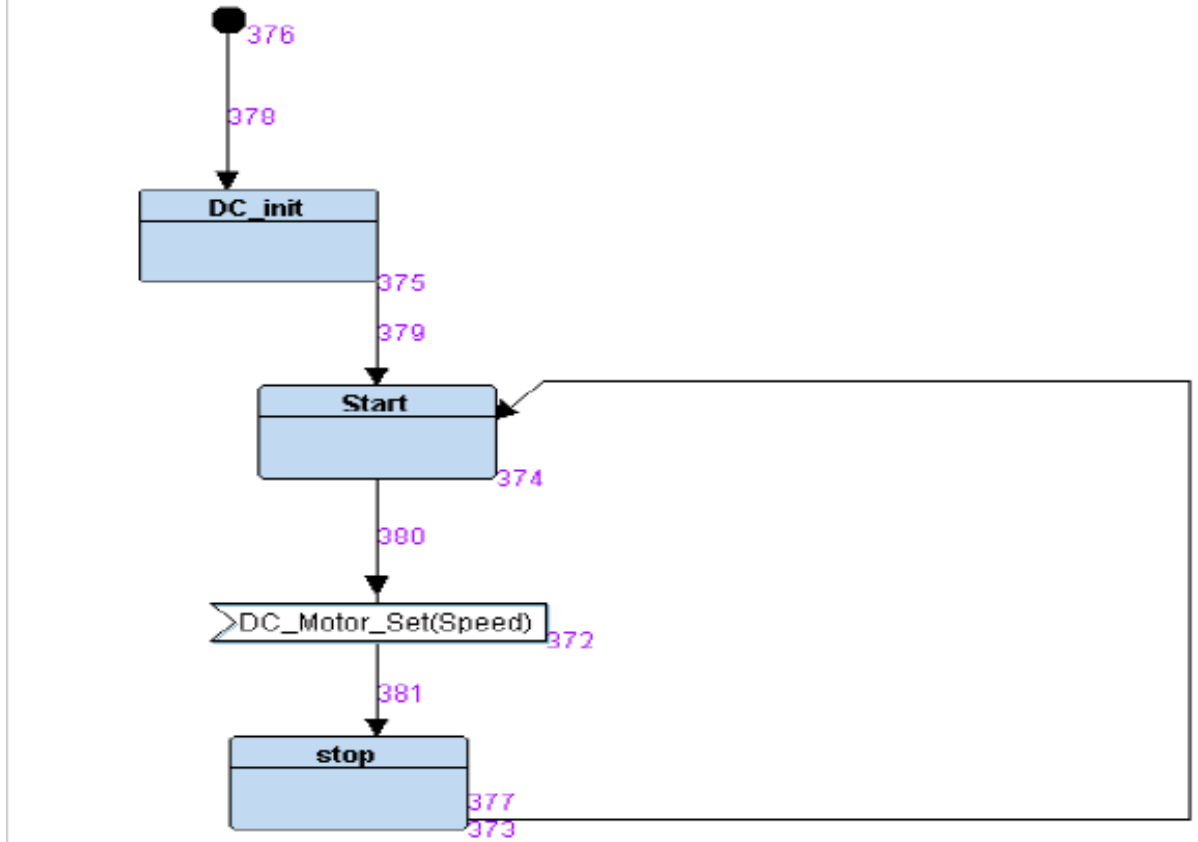
Modules Interfaces



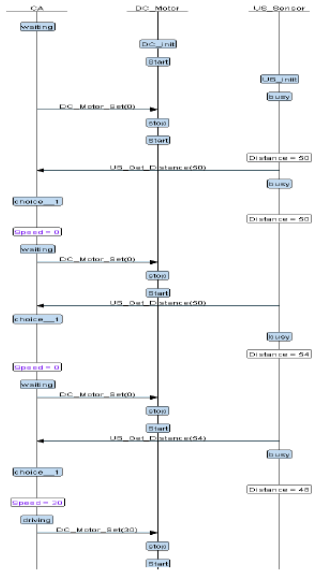
Sequence Diagram For CA



Sequence Diagram For Ultra Sonic



Sequence Diagram For CA



SW Logical Verification