Section 1 Description

Lab 3 let us design a small game in MSP430. In this game, I use Red square to represent the obstacles and blue square as bonus and yellow square as car. When the car hit the obstacle, it will game over but I designed 3 lives for the new game and when the car hit the bonus it will have 1 more life so when the life is zero the game is over and next screen will show the score after 3s. The screen will change to the main menu.

At the opening screen, it has the game name and my name. After 3s the screen will show the main menu and I can use joystick to change which sub screen I want to go. If I go to instruction screen. This screen will show the game instructions and I can Pushed the button 1 in order comeback to the main menu. If I choose the score history in main menu, the screen will show the highest score and if I pushed the bottom 1. It will back to main menu again. If I choose the Play game the screen will go to the game screen and it have life score and heist score show on the screen. I can move the car use the joystick and I can speed up the car if I push joystick up and slow down the speed if I pushed joystick down. In this screen when I pushed the bottom 1 it will show the pause screen and pause the game and I can choose whether I go back to main menu or continue the game.

Section 2 Finite State Machine Design

In the lab rubric only have two major FSM. But did not exactly follow it. In this part, I will describe all the FSM I designed in the program.

2.1 Main FSM:

I use this FSM to switch the screen between opening screen and main menu screen. This FSM have three state in the code show as follow. This FSM in my program is not play the most important role as it described in the rubric. I only use it at the most beginning of the program. The only thing it can do is draw main menu screen after the opening screen displayed after 3s.

void ScreensFSM()

{

    static enum states {INCEPTION, OPENING, MENU} state = INCEPTION;

    static OneShotSWTimer\_t OST;

    InitOneShotSWTimer(&OST,

                       TIMER32\_1\_BASE,

                       OPENING\_WAIT);

    // Set the default outputs

    bool drawOpening = false;

    bool drawMenu = false;

    switch (state)

    {

    case INCEPTION:

        StartOneShotSWTimer(&OST);

        // State transition

        state = OPENING;

        // The output(s) that are affected in this transition

        drawOpening = true;

        break;

    case OPENING:

        if (OneShotSWTimerExpired(&OST))

        {

            // State transition

            state = MENU;

            // The output(s) that are affected in this transition

            drawMenu = true;

        }

        break;

    case MENU:

        break;

    } // End of switch-case

    // Implement actions based on the outputs of the FSM

    if (drawOpening)

            DrawOpeningScreen();

    if (drawMenu)

        DrawMenuScreen();

}

2.2 Main Menu FSM

I call this FSM as Main menu FSM instead of Menu FSM because it only controls the main menu state change. So, it has three different state: Play game, how to play and Score History. When joystick moves will change the state and display in the Main menu screen. Show in the following code.

void MainMenuFSM()

{// choose which screen will showing up next

    typedef enum {PlayGame, HowToPlay, ScoreHistory} menu\_t; //states of what process going

    static menu\_t menuOption = PlayGame;

    switch(menuOption){

    case PlayGame:

        getSampleJoyStick(&vx, &vy);

        if (Booster\_Joystick\_Pushed())

          {

              DrawGameScreen();

              Ingame = true;

              GO = false;

              score = 0;

              ArrowMove = false;

              life = 3;

              bonus = 0;

          }

        if (ArrowMove)

        {

            if (vy > UP\_THRESHOLD)

            {

                JoystickBacktoMiddle();

                PrintMenuOption(History);

                menuOption = ScoreHistory;

            }

            if (vy < DOWN\_THRESHOLD)

            {

                JoystickBacktoMiddle();

                PrintMenuOption(Play);

                menuOption = HowToPlay;

            }

        }

    break;

    case HowToPlay:

        getSampleJoyStick(&vx, &vy);

        if (Booster\_Joystick\_Pushed())

         {

             DrawInstructionScreen();

             ArrowMove = false;

         }

        if(ArrowMove)

        {

                    if (vy > UP\_THRESHOLD)

                    {

                        JoystickBacktoMiddle();

                        PrintMenuOption(Game);

                        menuOption = PlayGame;

                    }

                    else if (vy < DOWN\_THRESHOLD)

                    {

                        JoystickBacktoMiddle();

                        PrintMenuOption(History);

                        menuOption = ScoreHistory;

                    }

        }

        else

        {

            if (Booster\_Top\_Button\_Pushed())

                 {

                        DrawMenuScreen();

                        PrintMenuOption(Play);

                        ArrowMove = true;

                 }

        }

    break;

    case ScoreHistory:

        getSampleJoyStick(&vx, &vy);

        if (Booster\_Joystick\_Pushed())

        {

            DrawScoreScreen();

            ArrowMove = false;

        }

        if (ArrowMove)

        {

            if (vy > UP\_THRESHOLD)

            {

                JoystickBacktoMiddle();

                PrintMenuOption(Play);

                menuOption = HowToPlay;

            }

            else if (vy < DOWN\_THRESHOLD)

            {

                JoystickBacktoMiddle();

                PrintMenuOption(Game);

                menuOption = PlayGame;

            }

        }

        else

        {

            if (Booster\_Top\_Button\_Pushed())

                 {

                DrawMenuScreen();

                PrintMenuOption(History);

                ArrowMove = true;

                 }

        }

    break;

    }

}

2.3 Print Main Menu FSM

I just change the arrow position in each state to make my program easier to understand.

2.4 Pause Screen FSM

This FSM have the same function as Main Menu FSM does and the code is show as followed.

void PrintMenuOption(S)

{

    switch(S){

    case Play:

        ClearString("->",2,1);

        ClearString("->",6,1);

        PrintString("->",4,1);

    break;

    case History:

        ClearString("->",2,1);

        ClearString("->",4,1);

        PrintString("->",6,1);

    break;

    case Game:

        ClearString("->",4,1);

        ClearString("->",6,1);

        PrintString("->",2,1);

    break;

    }

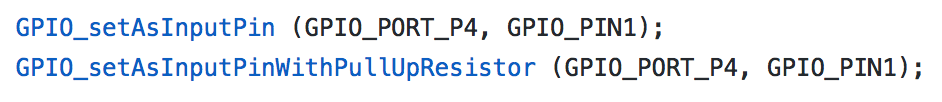
}

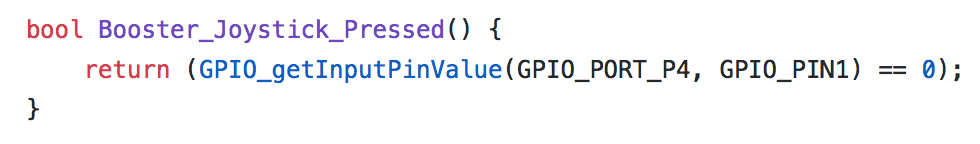
Section 3: HAL design

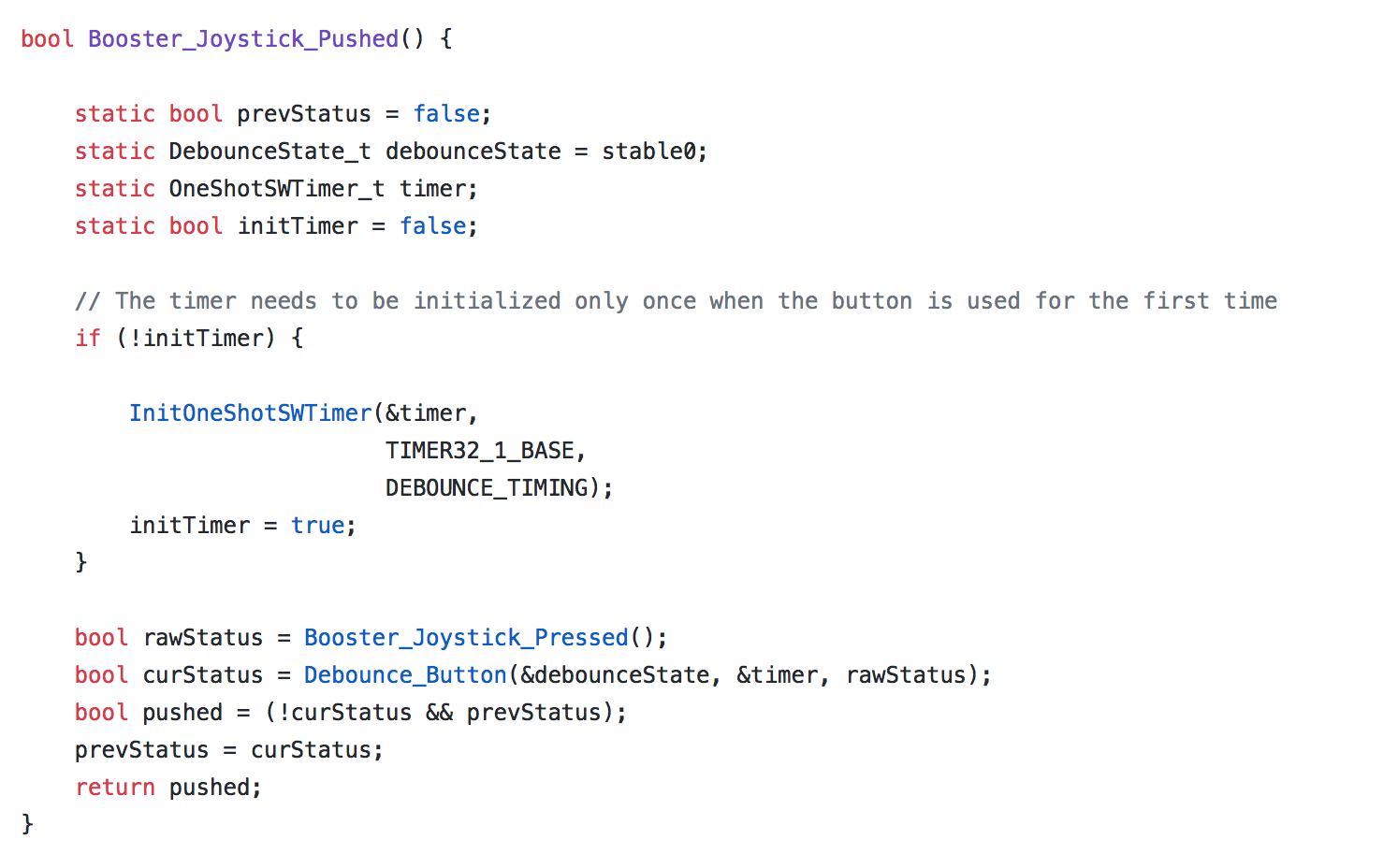
I include the ADC file given in the homework 5 part 4. The most important function in this design let us determine the joystick position.

For the bottom HAL I add the function to determine whether joystick been pressed or pushed.

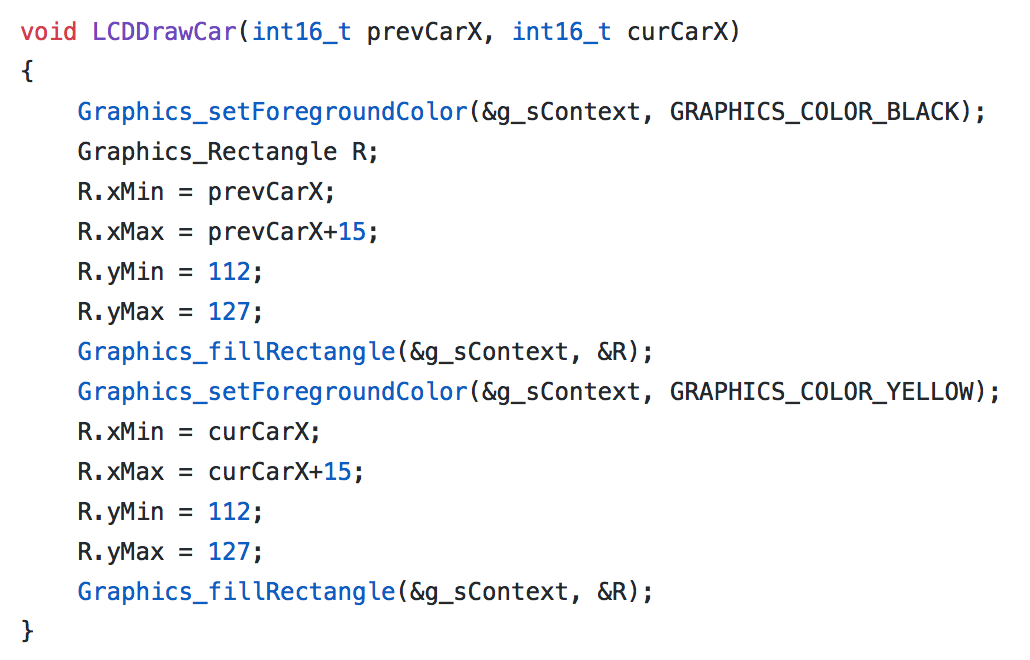
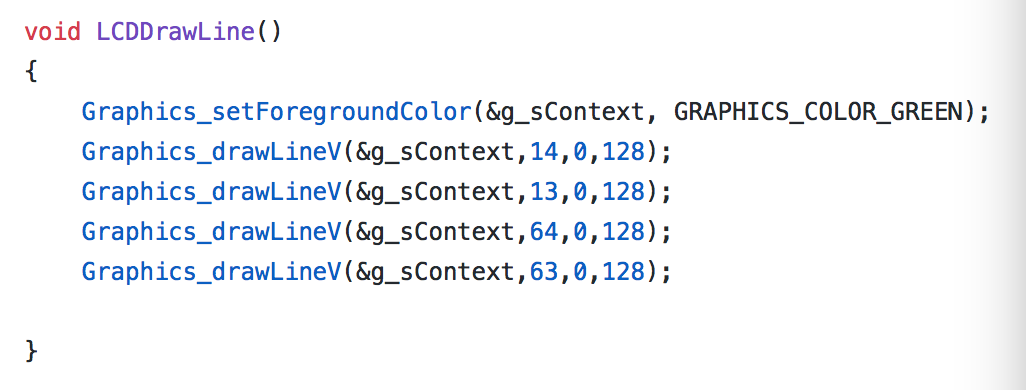
And I set the Joystick in the initial bouton function show as follow.



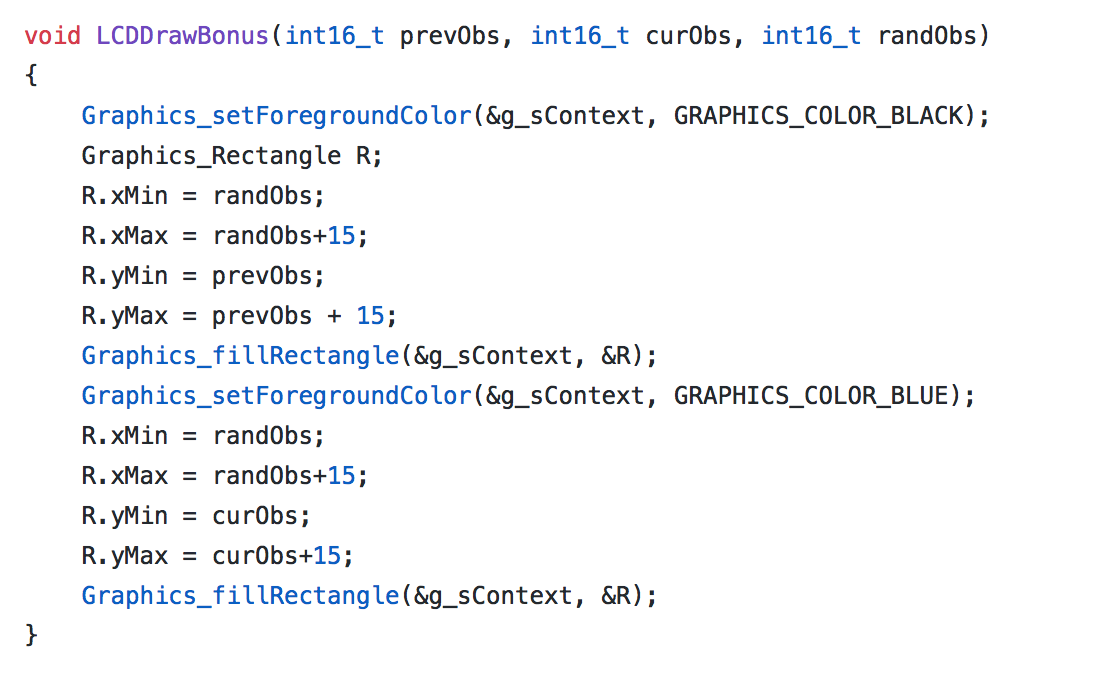


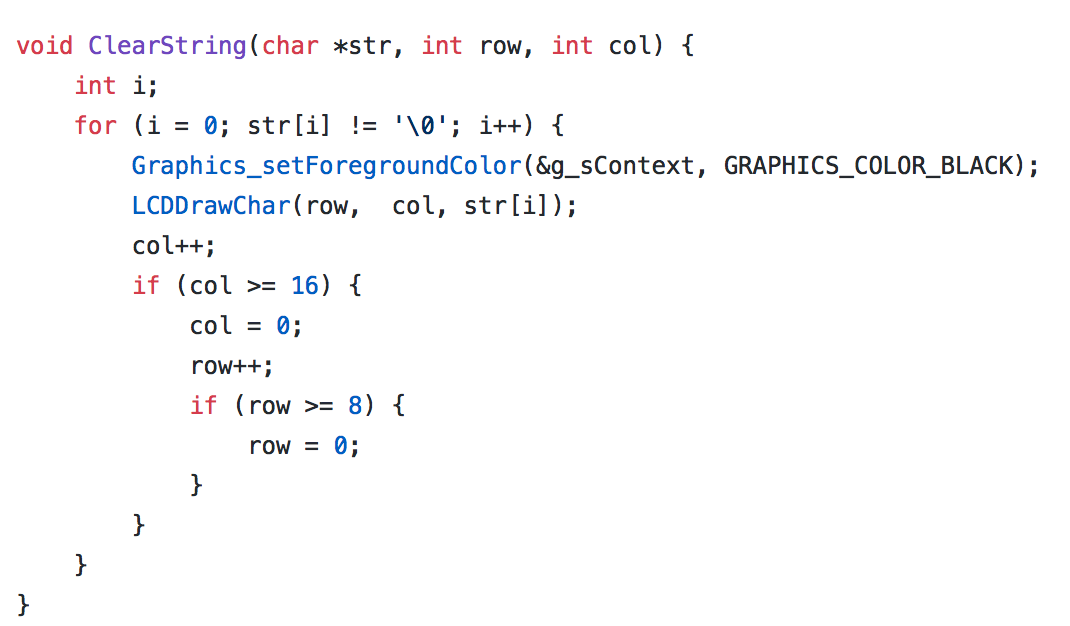


For the display HAL I add code to draw the road, the car, the bonus, and clear string as follow.









I put my code in main.c to finished the main function. Show as following. And my program has the function that generate the bonus randomly and when car hit it increased the life of the car. And when the car hit the bonus it will increase the live and when the car hit the obstacle it will decrease the life when life all gone it will show game over, and 3s later it will back to main menu.

void MoveObs()

{

    if(curObs1 == -1)

    {

        if(curObs3 == -1 || curObs3 > 50)

        {

            obs0 = randObsX();

            curObs1 = 0;

            prevObs1 = 0;

            hit1 = true;

        }

    }

    else if(curObs1 < 128 && curObs1 > -1)

    {

        curObs1 += acc;

        LCDDrawObs(prevObs1, curObs1, obs0);

        prevObs1 = curObs1;

    }

    else if(curObs1 >= 128)

    {

        curObs1 = -1;

        bonus--;

        score++;

    }

    if(curObs2 == -1)

    {

        if(curObs1 > 50)

        {

            obs1 = randObsX();

            curObs2 = 0;

            prevObs2 = 0;

            hit2 = true;

        }

    }

    else if(curObs2 < 128 && curObs2 > -1)

    {

        curObs2 += acc;

        LCDDrawObs(prevObs2, curObs2, obs1);

        prevObs2 = curObs2;

    }

    else if(curObs2 >= 128)

    {

        curObs2 = -1;

        bonus--;

        score++;

    }

    if(curObs3 == -1)

        {

            if(curObs2 > 50)

            {

                obs2 = randObsX();

                curObs3 = 0;

                prevObs3 = 0;

                hit3 = true;

            }

        }

        else if(curObs3 < 128 && curObs3 > -1)

        {

            curObs3 += acc;

            LCDDrawObs(prevObs3, curObs3, obs2);

            prevObs3 = curObs3;

        }

        else if(curObs3 >= 128)

        {

            curObs3 = -1;

            bonus--;

            score++;

        }

}

void MoveCar()

{

    getSampleJoyStick(&vx, &vy);

    if (vx > UP\_THRESHOLD)

    {

        curCarX +=acc;

        if(prevCarX < 47)

        {

            LCDDrawCar(prevCarX, curCarX);

            prevCarX = curCarX;

        }

        else

        {

            curCarX = 47;

            LCDDrawCar(prevCarX, curCarX);

            prevCarX = curCarX;

        }

    }

    if (vx < DOWN\_THRESHOLD)

    {

        curCarX -= acc;

        if(prevCarX > 15)

        {

            LCDDrawCar(prevCarX, curCarX);

            prevCarX = curCarX;

        }

        else

        {

            curCarX=15;

            LCDDrawCar(prevCarX, curCarX);

            prevCarX = curCarX;

        }

    }

}

void Evaluate()

{

    if(curObs1 >= 97 && curObs1 <= 127)

    {

        if(obs0 > curCarX + 15 || curCarX > obs0 + 15)

        {

        }

        else

        {

            if(hit1)

            {

                life--;

                hit1 = false;

            }

        }

    }

    if(curObs2 >= 97 && curObs2 <= 127)

        {

            if(obs1 > curCarX + 15 || curCarX > obs1 + 15)

            {

            }

            else

            {

                if(hit2)

                {

                    life--;

                    hit2 = false;

                }

            }

        }

    if(curObs3 >= 97 && curObs3 <= 127)

        {

            if(obs2 > curCarX + 15 || curCarX > obs2 + 15)

            {

            }

            else

            {

                if(hit3)

                {

                    life--;

                    hit3 = false;

                }

            }

        }

    if(curBonus >=97 && curBonus <=127)

    {

        if(bonusX > curCarX + 15 || curCarX > bonusX + 15)

        {

        }

        else

        {

            curBonus = 128;

            LCDDrawBonus(prevBonus, curBonus, bonusX);

            curBonus = -1;

            life++;

        }

    }

    if(life <= 0)

    {

        GO = true;

    }

}

void MoveBonus()

{

    if(curBonus == -1)

    {

        if(bonus <= 0)

        {

            if((curObs1 >15 || curObs1 == -1) && (curObs2 >15 || curObs2 == -1) && (curObs3 >15 || curObs3 == -1))

            {

                bonusX = randObsX();

                curBonus = 0;

                prevBonus = 0;

                bonus = rand()%5;

            }

        }

    }

    else if(curBonus < 128 && curBonus > -1)

    {

        curBonus += acc;

        LCDDrawBonus(prevBonus, curBonus, bonusX);

        prevBonus = curBonus;

    }

    else if(curBonus >= 128)

    {

        curBonus = -1;

    }

}

I use the if loop, while loop to make the game work as the lab rubric stated and the main function you can see as followed:

int main(void) {

    WDT\_A\_hold(WDT\_A\_BASE);

    BSP\_Clock\_InitFastest();

    initADC\_Multi();

    InitGraphics();

    initJoyStick();

    InitHWTimers();

    InitButtons();

    InitLEDs();

    startADC();

    static OneShotSWTimer\_t CAR;

    bool swTimerExpired;

    while (1)

    {

        // Do not delete this statement. We will use this function to check if your program does not block on anything.

        if (Booster\_Bottom\_Button\_Pushed())

            Toggle\_Launchpad\_Right\_Red\_LED();

        ScreensFSM();

        if (Ingame)

        {

            if (Booster\_Top\_Button\_Pushed())

                 {

                    DrawPauseScreen();

                    pause = true;

                 }

            if(pause)

            {

                PauseScreenFSM();

            }

            else

            {

                InitOneShotSWTimer(&CAR, TIMER32\_1\_BASE, CAR\_WAIT);

                StartOneShotSWTimer(&CAR);

                swTimerExpired = OneShotSWTimerExpired(&CAR);

                while (!swTimerExpired)

                    {

                        swTimerExpired = OneShotSWTimerExpired(&CAR);

                    }

                MoveCar();

                MoveObs();

                MoveBonus();

                Evaluate();

                char buf = ((score/10)%10)+'0';

                PrintString(&buf, 1, 8);

                buf = (score%10)+'0';

                PrintString(&buf, 1, 9);

                buf = (life%10)+'0';

                PrintString(&buf, 7, 8);

                getSampleJoyStick(&vx, &vy);

                if (vy > UP\_THRESHOLD)

                {

                    acc ++;

                }

                else if (vy < DOWN\_THRESHOLD)

                {

                    if(acc > 1)

                    {

                        acc--;

                    }

                }

            }

            if(GO)

            {

                DrawGameOverScreen();

                DrawMenuScreen();

                PrintMenuOption(Game);

                ArrowMove = true;

                Ingame = false;

            }

        }

        else

         {

            MainMenuFSM();

            curObs1 = -1;

            curObs2 = -1;

            curObs3 = -1;

            prevCarX = 15;

            curCarX = 15;

            acc = 1;

         }

    }

}