Lab7 code

```c
//pingpong template file for Lab #7
//Revised by Jianjian Song to add pressing early penalty
//June 2019
#include "xparameters.h"
#include "xgpio.h"
#include "led_ip.h"
// Include scutimer header file
#include "XScuTimer.h"
//====================================================
XScuTimer Timer; /* Cortex A9 SCU Private Timer Instance */
int delay(int i);
void MoveBallRight(void);
void MoveBallLeft(void);

#define ONE_TENTH 32500000 // half of the CPU clock speed/10
#define START 1
#define STOP 0
#define LEFT 0
#define RIGHT 1
#define RESETBUTTON 0b0100
#define STARTBUTTON 0b0010
#define LEFTPADDLE 0b1000
#define RIGHTPADDLE 0b0001

#define LED_PATTERNS_ORDER_LEFT_OUT -1
#define LED_PATTERNS_ORDER_RIGHT_OUT 6

#define RUNKEEP 1
#define RUNFOBBIDEN 0

int psb_check, dip_check, LedState, Status, dip_check_prev, psb_check_prev;


XGpio dip, push;

// PS Timer related definitions
XScuTimer_Config *ConfigPtr;
XScuTimer *TimerInstancePtr = &Timer;

int LED_PATTERNS[6] = { 0b0000, 0b1000, 0b0100, 0b0010, 0b0001, 0b0000 };
int LED_PATTERNS_ORDER[6] = {0,1,2,3,4,5};
int scoreright, scoreleft;
```

```c
char GameOver, StartDirection;

int main(void) {

    //initialize variables, timers, ports
    XGpio_Initialize(&dip, XPAR_SWITCHES_DEVICE_ID);
    XGpio_SetDataDirection(&dip, 1, 0xffffffff);

    XGpio_Initialize(&push, XPAR_BOTTONS_DEVICE_ID);
    XGpio_SetDataDirection(&push, 1, 0xffffffff);

    //use psb_check_prev and psb_check to check if the button has been push

    psb_check_prev = XGpio_DiscreteRead(&push, 1);

    // Initialize the timer
    ConfigPtr = XScuTimer_LookupConfig(XPAR_PS7_SCUTIMER_0_DEVICE_ID);
    Status = XScuTimer_CfgInitialize(TimerInstancePtr, ConfigPtr,
            ConfigPtr->BaseAddr);
    if (Status != XST_SUCCESS) {
        xil_printf("Timer init() failed\r\n");
        return XST_FAILURE;
    }

    // Read dip switch values
    dip_check_prev = XGpio_DiscreteRead(&dip, 1);
    // Load timer with delay in multiple of ONE_TENTH
    XScuTimer_LoadTimer(TimerInstancePtr, ONE_TENTH*dip_check_prev);
    // Set AutoLoad mode
    XScuTimer_EnableAutoReload(TimerInstancePtr);
    // Start the timer
    XScuTimer_Start(TimerInstancePtr);

    xil_printf("-- Start of the Ping Pong Program --\r\n");
    GameOver = STOP;
    scoreright = 0;
    scoreleft = 0;
    xil_printf("Score Left = %d   Score Right = %d\r\n", scoreright, scorele
ft);
    StartDirection = LEFT;
    while (1) {
        // Read push buttons and reset score if Button 2 is pressed
        psb_check = XGpio_DiscreteRead(&push, 1);
        if (psb_check == RESETBUTTON && psb_check != psb_check_prev)//reset game
        {
            xil_printf("\n\rNew Game - Scores Reset\r\n");
            scoreright = 0;
            scoreleft = 0;
            xil_printf("Score Left = %d   Score Right = %d\r\n", scoreright,
                    scoreleft);
            psb_check_prev = psb_check;
        }

        //check the STARTBUTTON been pushed
        if (psb_check == STARTBUTTON) {
            GameOver = START;   //start game
            psb_check_prev = psb_check;
        }

        //check the game status,if start the ball will move right or right
        if (GameOver == STOP) {
            if (StartDirection == LEFT) {
                LED_IP_mWriteReg(XPAR_LED_IP_0_S00_AXI_BASEADDR, 0,
                        LED_PATTERNS[5]);
            } else {
                LED_IP_mWriteReg(XPAR_LED_IP_0_S00_AXI_BASEADDR, 0,
                        LED_PATTERNS[0]);
            }
        } else {
            if (StartDirection) {
                MoveBallLeft();
            } else {
                MoveBallRight();
            }
        }


    }
} //main()

void MoveBallRight(void) {
    int led_order;
    int run;

    //keep the score add once
    int getOut = 1;

    for (led_order = LED_PATTERNS_ORDER[5]; led_order >= LED_PATTERNS_ORDER[0];) {

        run = RUNKEEP;
        while (run) {
            //check the switches changed
            dip_check = XGpio_DiscreteRead(&dip, 1);
            if (dip_check != dip_check_prev) {
                xil_printf("DIP Switch Status %x, %x\r\n", dip_check_prev,
                        dip_check);
                dip_check_prev = dip_check;
                // load timer with the new switch settings
                XScuTimer_LoadTimer(TimerInstancePtr, ONE_TENTH*dip_check);
            }

            //check the LEFTPADDLE pushed
            psb_check = XGpio_DiscreteRead(&push, 1);
            if (psb_check == LEFTPADDLE) {
                //set StartDirection
                if (led_order == LED_PATTERNS_ORDER[1]) {
                    StartDirection = RIGHT;
                    run = RUNFOBBIDEN;
                    led_order = LED_PATTERNS_ORDER_LEFT_OUT;
                } else {
                    GameOver = STOP;
                    LED_IP_mWriteReg(XPAR_LED_IP_0_S00_AXI_BASEADDR, 0,
                            LED_PATTERNS[0]);
                    StartDirection = RIGHT;
                    scoreleft += 1;
                    run = RUNFOBBIDEN;
                    led_order = LED_PATTERNS_ORDER_LEFT_OUT;
                    LED_IP_mWriteReg(XPAR_LED_IP_0_S00_AXI_BASEADDR, 0,
                            LED_PATTERNS[led_order]);
                    xil_printf("Score Left = %d   Score Right = %d\r\n", scoreright, scoreleft);
                }
            } else {
                //set GameOver; display scores
                if (led_order == LED_PATTERNS_ORDER[0] && getOut) {
                    scoreleft += 1;
                    GameOver = STOP;
                    StartDirection = RIGHT;

                    xil_printf("Score Left = %d   Score Right = %d\r\n",
                            scoreright, scoreleft);
                    run = RUNFOBBIDEN;
                    getOut = 0;
                }

                if (XScuTimer_IsExpired(TimerInstancePtr)) {
                    // clear status bit
                    XScuTimer_ClearInterruptStatus(TimerInstancePtr);
                    led_order--;
                    LED_IP_mWriteReg(XPAR_LED_IP_0_S00_AXI_BASEADDR, 0,
                            LED_PATTERNS[led_order]);
                    run = RUNFOBBIDEN;
                }
            }
        }

    }
}

void MoveBallLeft(void) {
    int led_order;
    int run;

    //keep the score add once
    int getOut = 1;
    for (led_order = LED_PATTERNS_ORDER[1]; led_order <= LED_PATTERNS_ORDER[5];) {
        run = RUNKEEP;
        while (run) {
            dip_check = XGpio_DiscreteRead(&dip, 1);
            if (dip_check != dip_check_prev) {
                xil_printf("DIP Switch Status %x, %x\r\n", dip_check_prev,
                        dip_check);
                dip_check_prev = dip_check;
                // load timer with the new switch settings
                XScuTimer_LoadTimer(TimerInstancePtr, ONE_TENTH*dip_check);

            }
            psb_check = XGpio_DiscreteRead(&push, 1);
            if (psb_check == RIGHTPADDLE) {
                //set StartDirection
                if (led_order == LED_PATTERNS_ORDER[4]) {
                    StartDirection = LEFT;
                    run = 0;
                    led_order = LED_PATTERNS_ORDER_RIGHT_OUT;
                } else {
                    GameOver = STOP;
                    StartDirection = LEFT;
                    scoreright += 1;
                    run = RUNFOBBIDEN;

                    led_order = LED_PATTERNS_ORDER_RIGHT_OUT;
                    LED_IP_mWriteReg(XPAR_LED_IP_0_S00_AXI_BASEADDR, 0,
                            LED_PATTERNS[led_order]);

                    xil_printf("Score Left = %d   Score Right = %d\r\n", scoreright, scoreleft);
                }

            } else {
                if (led_order == LED_PATTERNS_ORDER[5] && getOut) {
                    //set GameOver; display scores
                    scoreright += 1;
                    GameOver = STOP;
                    StartDirection = LEFT;
                    xil_printf("Score Left = %d   Score Right = %d\r\n",
                            scoreright, scoreleft);
                    run = RUNFOBBIDEN;
                    getOut = 0;
                }

                if (XScuTimer_IsExpired(TimerInstancePtr)) {
                    // clear status bit
                    XScuTimer_ClearInterruptStatus(TimerInstancePtr);
                    led_order++;
                    LED_IP_mWriteReg(XPAR_LED_IP_0_S00_AXI_BASEADDR, 0,
                            LED_PATTERNS[led_order]);
                    run = RUNFOBBIDEN;
                }
            }
        }
    }
}
```