1. 游戏总体介绍

1.1 游戏进程

启动游戏后首先进入初始化界面，播放初始化动画与音乐，流水灯启动，之后进入 PVP 模式状态，两个玩家操作对战，当有一方 HP 减至 0 则进入结束界面，播放结束动画与音乐。

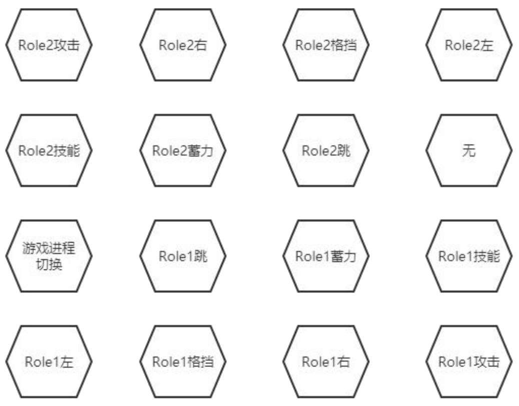
1.2 游戏对局规则

PVP，玩家 1 与玩家 2 利用各种攻击方式，率先将敌方血量减至 0 则获胜.

1.3 角色细节

玩家 1 的法术形式攻击为火球，伤害较高，跳跃灵敏，回复能力较弱，近身攻击伤害较低，平移能力较弱。玩家 2 的法术形式攻击为冰块，伤害较低，但远程坠落的冰块可对敌人造成硬控，跳跃灵敏，回复能力较弱，近战伤害高，平移能力较强，跳跃能力弱，回复能力适中。

1.4 游戏操作细节



左：角色左移

右：角色右移

格挡：吸收远程攻击法球，回复一定血量

攻击：在蓄力状态下可以发出水平法球，（此时冰块与火球都为伤害型攻击）蓄力时间越长，法球移速越快，

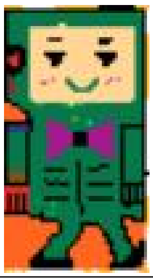
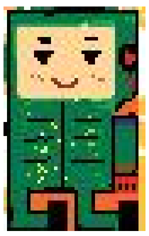
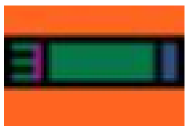
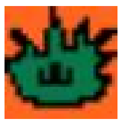
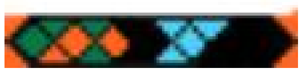
攻击距离越远，未蓄力时为近战攻击，近战攻击分为四档，可储存近战攻击次数，伤害随着档位提高也随之大幅提高，第四段普通攻击距离更远；当玩家在 2s 内被连续近身攻击打中两次以上时，进入受伤硬控状态，无法进行按键交互

跳：跳跃，跳跃至空中时会有 0.4s 的滞空状态，在滞空状态可以进行其他动作，也可以继续跳，当处于滞空状态未进行任何操作时，会在 0.4s 迅速落回地面。

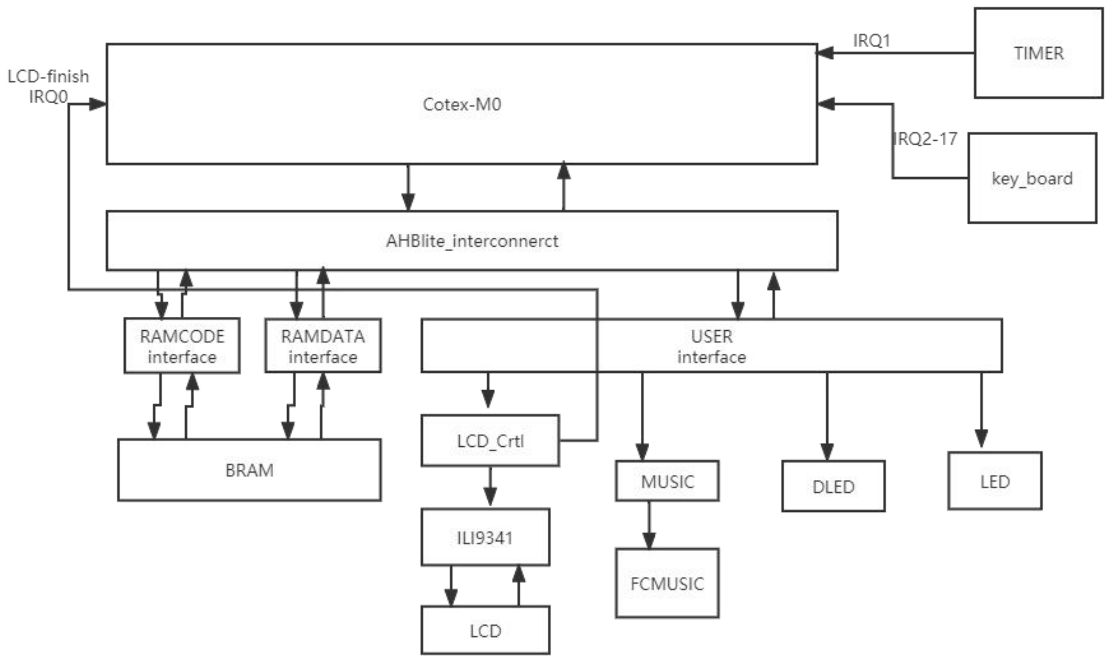
蓄力：法球的先决动作判定，蓄力与攻击或者技能按键间隔决定法球水平位移

技能：在蓄力状态下按下有效，在角色上空召唤法球攻击（冰块的攻击效果为控制，火球的攻击效果为大额伤害）

1.5 游戏图像设计

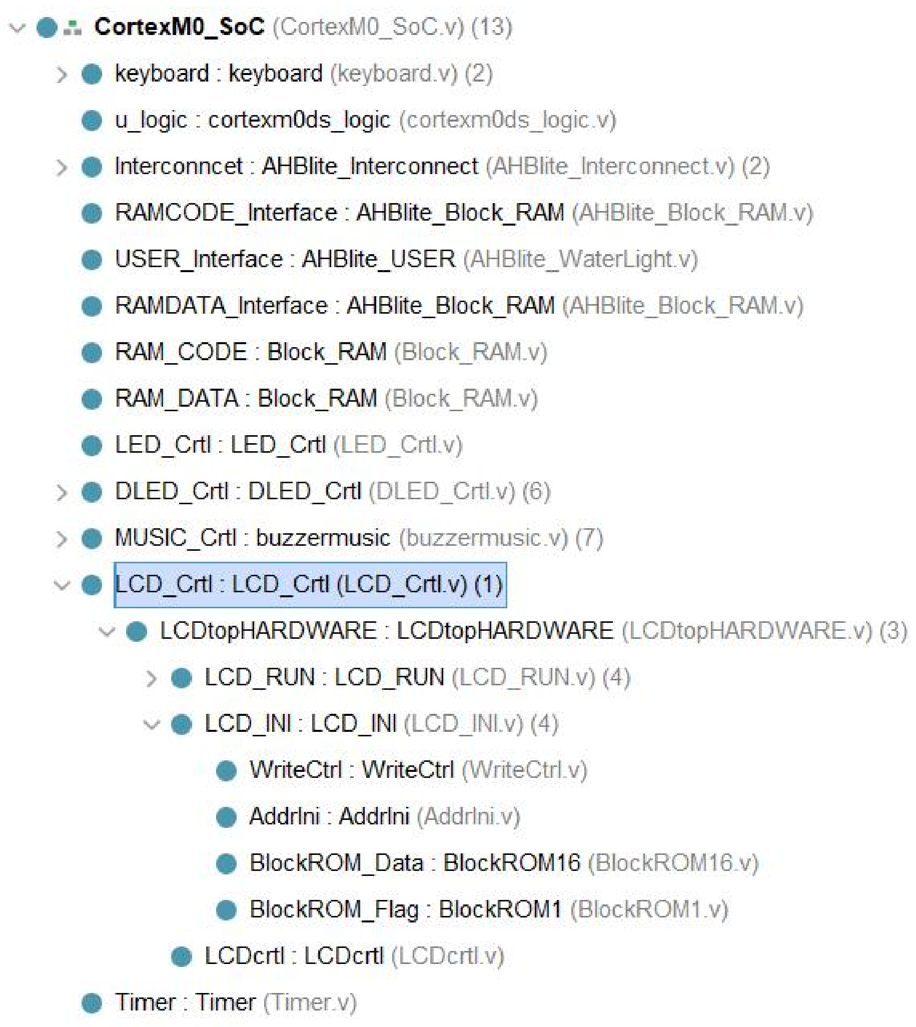
2. Soc 系统框图



TIMER
IRQ1
LCD-finish
IRQ0
Cotex-M0
IRQ2-17
key_board
AHBlite_interconnerct
RAMCODE interface
RAMDATA interface
USER interface
BRAM
LCD_Crtl
MUSIC
DLED
LED
ILI9341
FCMUSIC
LCD

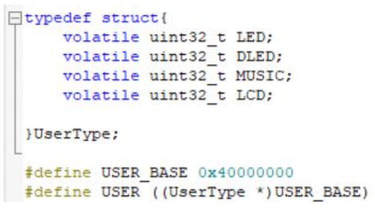
3. 硬件设计

工程结构：

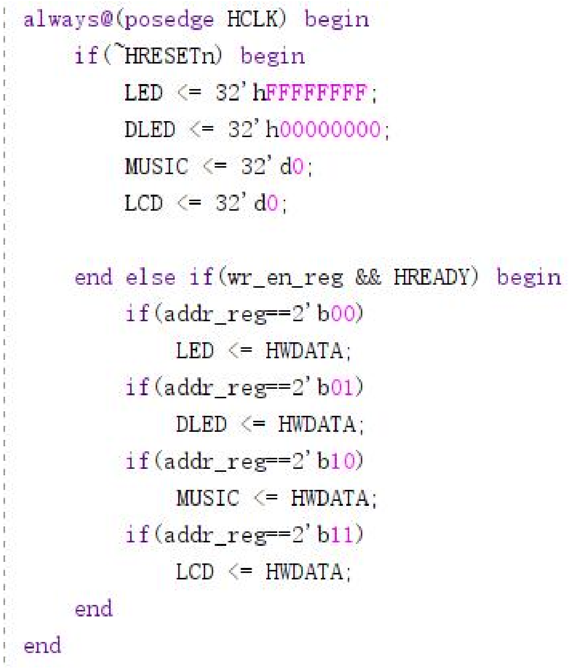


```
CortexM0_SoC (CortexM0_SoC.v) (13)
    keyboard : keyboard (keyboard.v) (2)
    u_logic : cortexm0ds_logic (cortexm0ds_logic.v)
    Interconncet : AHBlite_Interconnect (AHBlite_Interconnect.v) (2)
    RAMCODE_Interface : AHBlite_Block_RAM (AHBlite_Block_RAM.v)
    USER_Interface : AHBlite_USER (AHBlite_WaterLight.v)
    RAMDATA_Interface : AHBlite_Block_RAM (AHBlite_Block_RAM.v)
    RAM_CODE : Block_RAM (Block_RAM.v)
    RAM_DATA : Block_RAM (Block_RAM.v)
    LED_Crtl : LED_Crtl (LED_Crtl.v)
    DLED_Crtl : DLED_Crtl (DLED_Crtl.v) (6)
    MUSIC_Crtl : buzzermusic (buzzermusic.v) (7)
    LCD_Crtl : LCD_Crtl (LCD_Crtl.v) (1)
        LCDtopHARDWARE : LCDtopHARDWARE (LCDtopHARDWARE.v) (3)
            LCD_RUN : LCD_RUN (LCD_RUN.v) (4)
            LCD_INI : LCD_INI (LCD_INI.v) (4)
                WriteCtrl : WriteCtrl (WriteCtrl.v)
                AddrIni : AddrIni (AddrIni.v)
                BlockROM_Data : BlockROM16 (BlockROM16.v)
                BlockROM_Flag : BlockROM1 (BlockROM1.v)
            LCDcrtl : LCDcrtl (LCDcrtl.v)
    Timer : Timer (Timer.v)
```

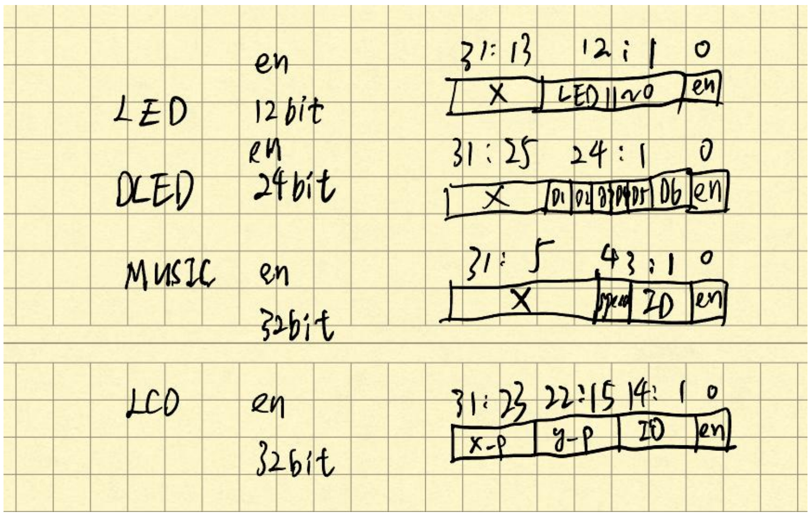
3.1 外设接口：



```c
typedef struct{
    volatile uint32_t LED;
    volatile uint32_t DLED;
    volatile uint32_t MUSIC;
    volatile uint32_t LCD;

}UserType;

#define USER_BASE 0x40000000
#define USER ((UserType *)USER_BASE)
```

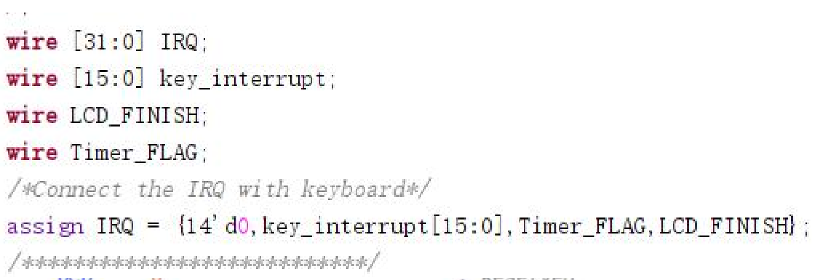


```
always@(posedge HCLK) begin
    if(~HRESETn) begin
        LED <= 32'hFFFFFFFF;
        DLED <= 32'h00000000;
        MUSIC <= 32'd0;
        LCD <= 32'd0;

    end else if(wr_en_reg && HREADY) begin
        if(addr_reg==2'b00)
            LED <= HWDATA;
        if(addr_reg==2'b01)
            DLED <= HWDATA;
        if(addr_reg==2'b10)
            MUSIC <= HWDATA;
        if(addr_reg==2'b11)
            LCD <= HWDATA;
    end
end
```

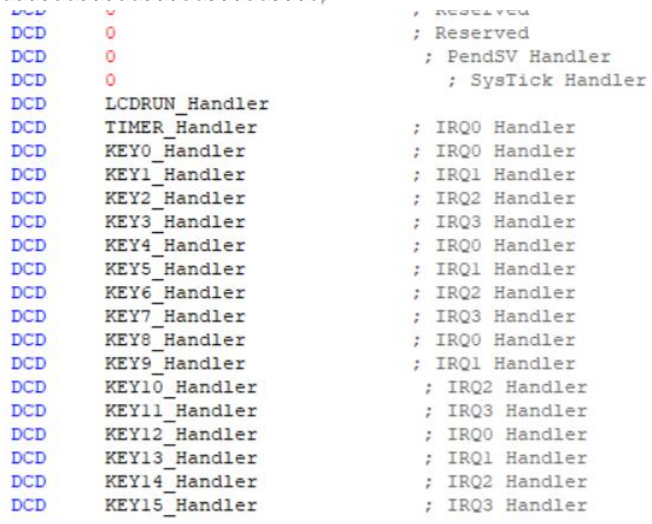


3.2 中断设计：

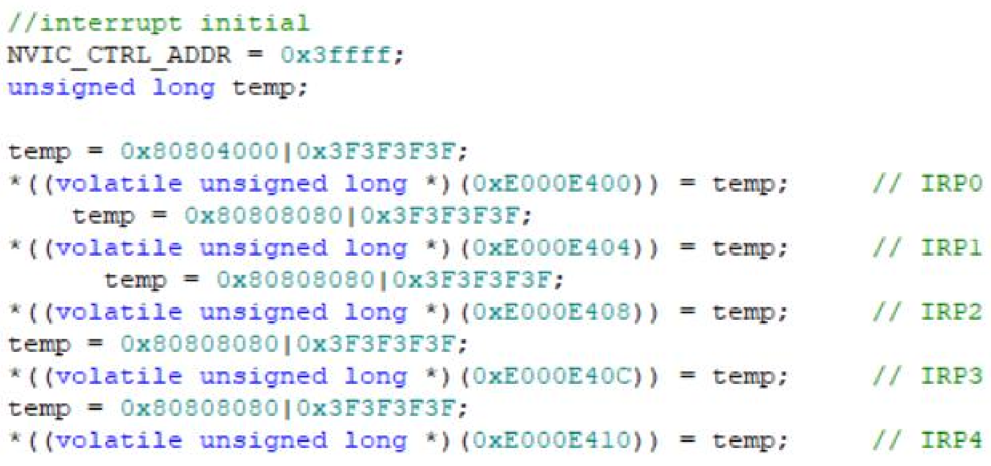
确保 LCD 中断的最高优先级，100ms 定时器次高优先级，按键中断最低优先级



```
wire [31:0] IRQ;
wire [15:0] key_interrupt;
wire LCD_FINISH;
wire Timer_FLAG;
/*Connect the IRQ with keyboard*/
assign IRQ = {14'd0,key_interrupt[15:0],Timer_FLAG,LCD_FINISH};
/****************************/
```



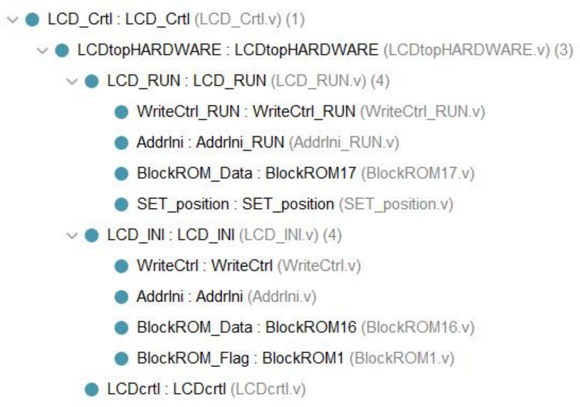
```
    DCD     0                           ; Reserved
    DCD     0                            ; PendSV Handler
    DCD     0                              ; SysTick Handler
    DCD     LCDRUN_Handler
    DCD     TIMER_Handler               ; IRQ0 Handler
    DCD     KEY0_Handler                ; IRQ0 Handler
    DCD     KEY1_Handler                ; IRQ1 Handler
    DCD     KEY2_Handler                ; IRQ2 Handler
    DCD     KEY3_Handler                ; IRQ3 Handler
    DCD     KEY4_Handler                ; IRQ0 Handler
    DCD     KEY5_Handler                ; IRQ1 Handler
    DCD     KEY6_Handler                ; IRQ2 Handler
    DCD     KEY7_Handler                ; IRQ3 Handler
    DCD     KEY8_Handler                ; IRQ0 Handler
    DCD     KEY9_Handler                ; IRQ1 Handler
    DCD     KEY10_Handler                ; IRQ2 Handler
    DCD     KEY11_Handler                ; IRQ3 Handler
    DCD     KEY12_Handler                ; IRQ0 Handler
    DCD     KEY13_Handler                ; IRQ1 Handler
    DCD     KEY14_Handler                ; IRQ2 Handler
    DCD     KEY15_Handler                ; IRQ3 Handler
```



```
//interrupt initial
NVIC_CTRL_ADDR = 0x3ffff;
unsigned long temp;

temp = 0x80804000|0x3F3F3F3F;
*((volatile unsigned long *)(0xE000E400)) = temp;      // IRP0
    temp = 0x80808080|0x3F3F3F3F;
*((volatile unsigned long *)(0xE000E404)) = temp;      // IRP1
      temp = 0x80808080|0x3F3F3F3F;
*((volatile unsigned long *)(0xE000E408)) = temp;      // IRP2
temp = 0x80808080|0x3F3F3F3F;
*((volatile unsigned long *)(0xE000E40C)) = temp;      // IRP3
temp = 0x80808080|0x3F3F3F3F;
*((volatile unsigned long *)(0xE000E410)) = temp;      // IRP4
```

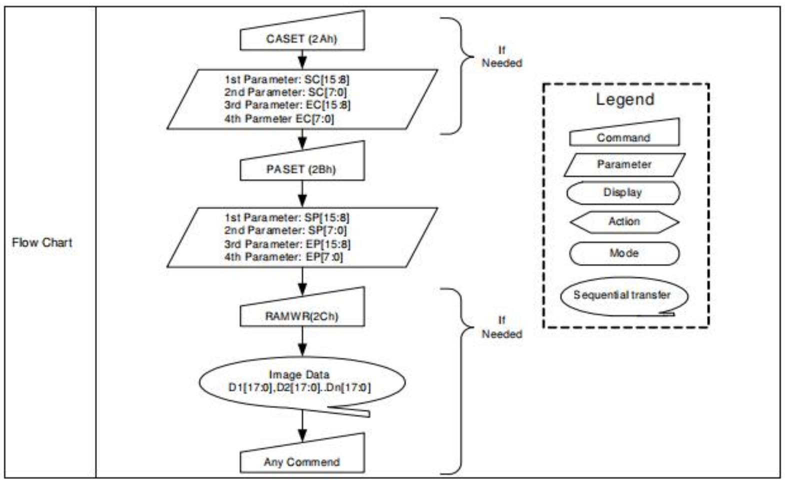
3.3 LCD 硬件加速：

Hdl 文件层次：



- LCD_Crtl : LCD_Crtl (LCD_Crtl.v) (1)
  - LCDtopHARDWARE : LCDtopHARDWARE (LCDtopHARDWARE.v) (3)
    - LCD_RUN : LCD_RUN (LCD_RUN.v) (4)
      - WriteCtrl_RUN : WriteCtrl_RUN (WriteCtrl_RUN.v)
      - AddrIni : AddrIni_RUN (AddrIni_RUN.v)
      - BlockROM_Data : BlockROM17 (BlockROM17.v)
      - SET_position : SET_position (SET_position.v)
    - LCD_INI : LCD_INI (LCD_INI.v) (4)
      - WriteCtrl : WriteCtrl (WriteCtrl.v)
      - AddrIni : AddrIni (AddrIni.v)
      - BlockROM_Data : BlockROM16 (BlockROM16.v)
      - BlockROM_Flag : BlockROM1 (BlockROM1.v)
    - LCDcrtl : LCDcrtl (LCDcrtl.v)

正常输出图片使用的芯片工作模式：



LCD 控制状态机：

```
module LCDcrtl(
    input clk,
    input rstn,
    input [30:0] LCD_CRTL,
    input run_finish,
    input ini_finish,
    input LCDcrtl_en,
    input POSITION_FINISH,
    output reg data_en,
    output reg position_en,
    output reg ini_en,
    output [7:0] run_addr_begin,
    output [7:0] run_addr_end,
    output reg LCD_MODE,
    output reg sel

    );

localparam  IDLE  = 4'd1;

localparam  WAIT_INI  = 4'd2;
localparam  INI  = 4'd3;
localparam  WAIT_CRTL  = 4'd4;
localparam  CRTL_POSITION  = 4'd5;
localparam  RUN  = 4'd6;
reg [3:0] cur_state, nxt_state;

always @ (*) begin
    case(cur_state)
    IDLE:nxt_state=WAIT_INI;
    WAIT_INI:nxt_state=INI;
    INI:nxt_state=ini_finish ? WAIT_CRTL:INI;
    WAIT_CRTL:nxt_state=LCDcrtl_en ? CRTL_POSITION:WAIT_CRTL;
    CRTL_POSITION:nxt_state=POSITION_FINISH ? RUN:CRTL_POSITION;
    RUN:nxt_state=run_finish ? WAIT_CRTL:RUN;
    default:nxt_state=IDLE;

    endcase
end

always @ (posedge clk or negedge rstn) begin
    if (~rstn) cur_state <= IDLE;
    else cur_state <= nxt_state;
end
```

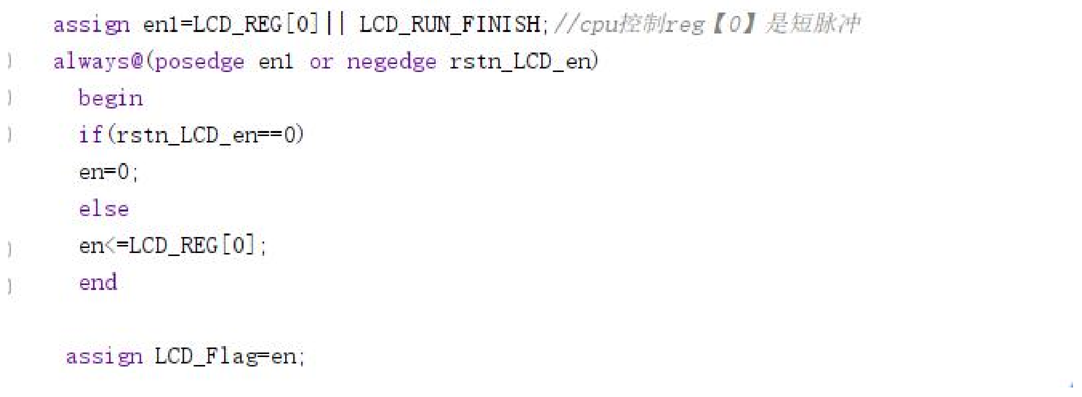
```
always @(posedge clk or negedge rstn) begin
    if (~rstn) begin

        ini_en<=0;
        LCD_MODE<=0;
        sel<=1;
        data_en<=0;
        position_en<=0;
    end else begin
        case (nxt_state)
            IDLE: begin

                ini_en<=0;
                LCD_MODE<=0;
                sel<=1;
                data_en<=0;
                position_en<=0;
            end
            WAIT_INI: begin

            end
            INI: begin

                ini_en<=1;
            end
            WAIT_CRTL: begin
                data_en<=0;
                position_en<=0;
                ini_en<=0;
                LCD_MODE<=1;
            end
            CRTL_POSITION:begin
                data_en<=0;
                ini_en<=0;

                sel<=1;
                position_en<=1;
            end
            RUN: begin
                position_en<=0;
                ini_en<=0;
                sel<=0;
                data_en<=1;
            end
            default: begin

            end
        endcase
    end
end
```

译码（示意部分代码）:

```
assign x_position_start=LCD_CRTL[30:23];
assign y_position_start=LCD_CRTL[22:14];
assign picture=LCD_CRTL[13:0];
```

```
always@(picture)
begin
case(picture)//jian ge 4 FFFF
14'd0:begin//左受伤2808
addr_begin=17'd0;
addr_end=17'd2807;
x_position_finish=x_position_start+8'd38;
y_position_finish=y_position_start+9'd71;

end
14'd1:begin//右受伤2808
addr_begin=17'd2812;
addr_end=17'd5619;
x position finish=x position start+8'd38;
```

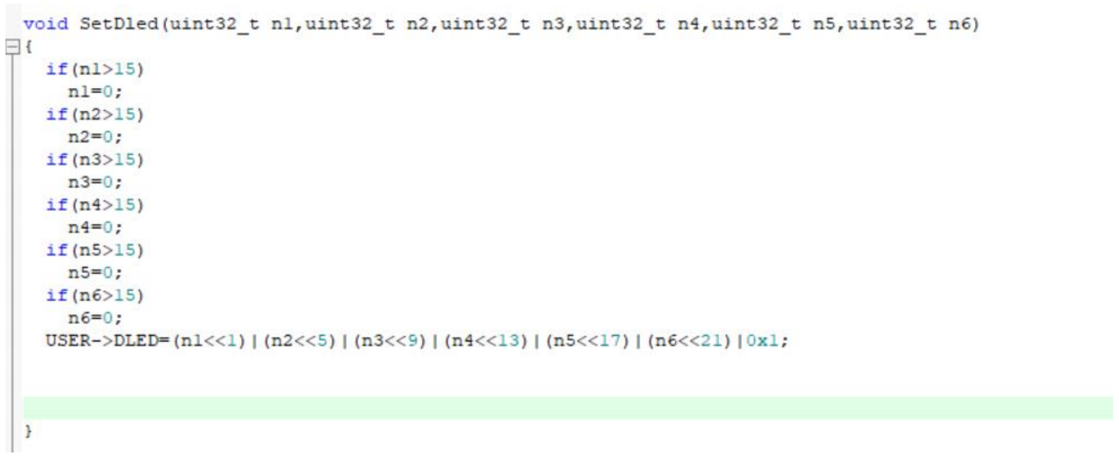
使能电路：（软件启动，硬件自动运行至结束，产生中断标志）



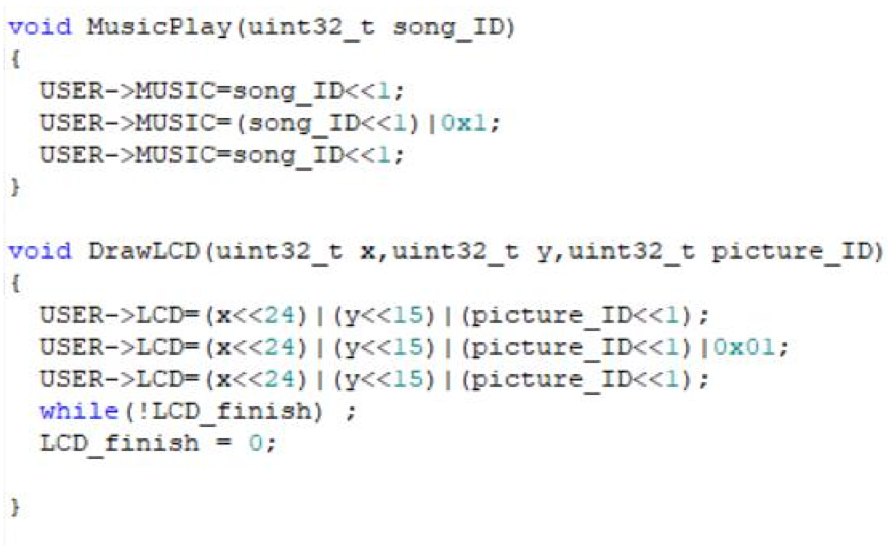
```
assign en1=LCD_REG[0]|| LCD_RUN_FINISH;//cpu控制reg【0】是短脉冲
always@(posedge en1 or negedge rstn_LCD_en)
  begin
  if(rstn_LCD_en==0)
  en=0;
  else
  en<=LCD_REG[0];
  end

 assign LCD_Flag=en;
```

4. 软件设计

4.1 寄存器处理函数

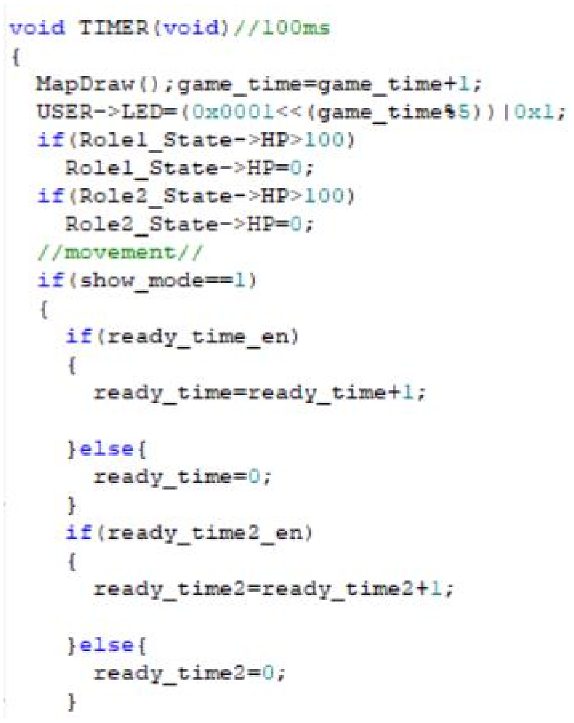


```
void SetDled(uint32_t n1,uint32_t n2,uint32_t n3,uint32_t n4,uint32_t n5,uint32_t n6)
{
  if(n1>15)
    n1=0;
  if(n2>15)
    n2=0;
  if(n3>15)
    n3=0;
  if(n4>15)
    n4=0;
  if(n5>15)
    n5=0;
  if(n6>15)
    n6=0;
  USER->DLED=(n1<<1)|(n2<<5)|(n3<<9)|(n4<<13)|(n5<<17)|(n6<<21)|0x1;


}
```

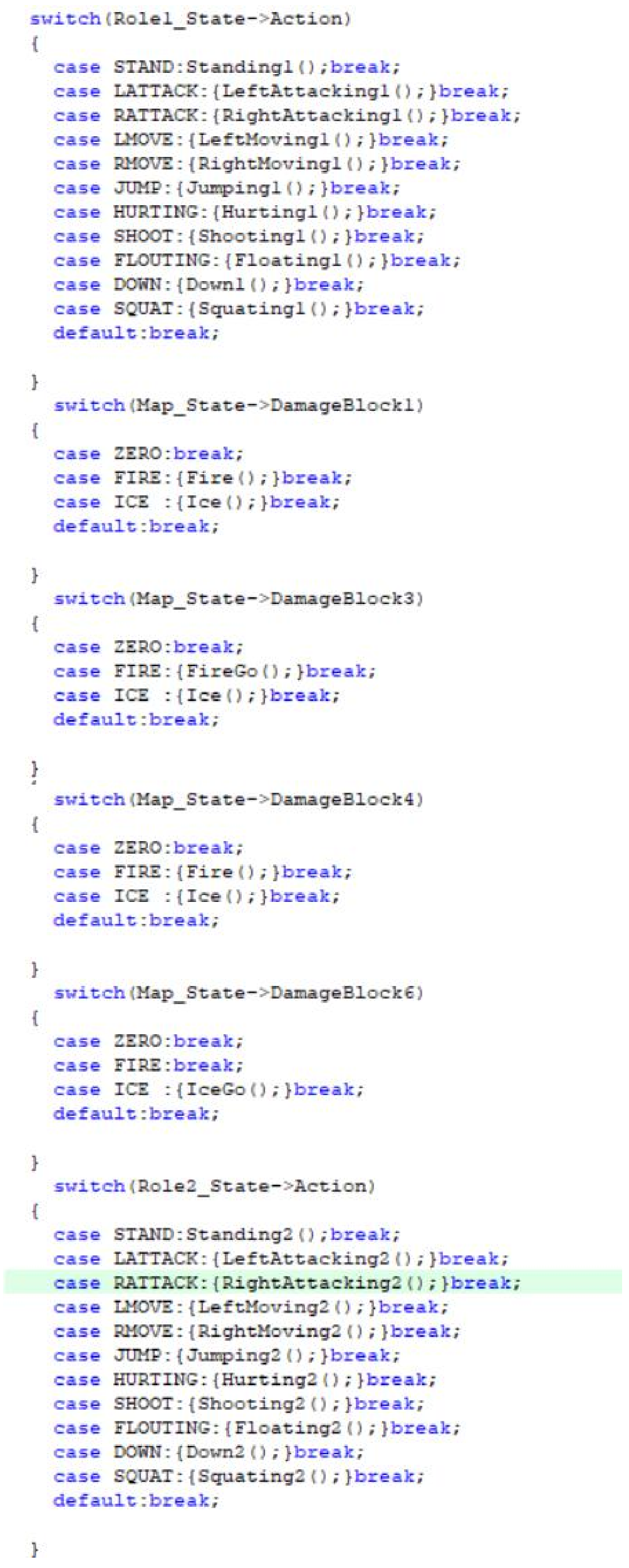


```
void MusicPlay(uint32_t song_ID)
{
  USER->MUSIC=song_ID<<1;
  USER->MUSIC=(song_ID<<1)|0x1;
  USER->MUSIC=song_ID<<1;
}

void DrawLCD(uint32_t x,uint32_t y,uint32_t picture_ID)
{
  USER->LCD=(x<<24)|(y<<15)|(picture_ID<<1);
  USER->LCD=(x<<24)|(y<<15)|(picture_ID<<1)|0x01;
  USER->LCD=(x<<24)|(y<<15)|(picture_ID<<1);
  while(!LCD_finish) ;
  LCD_finish = 0;

}
```

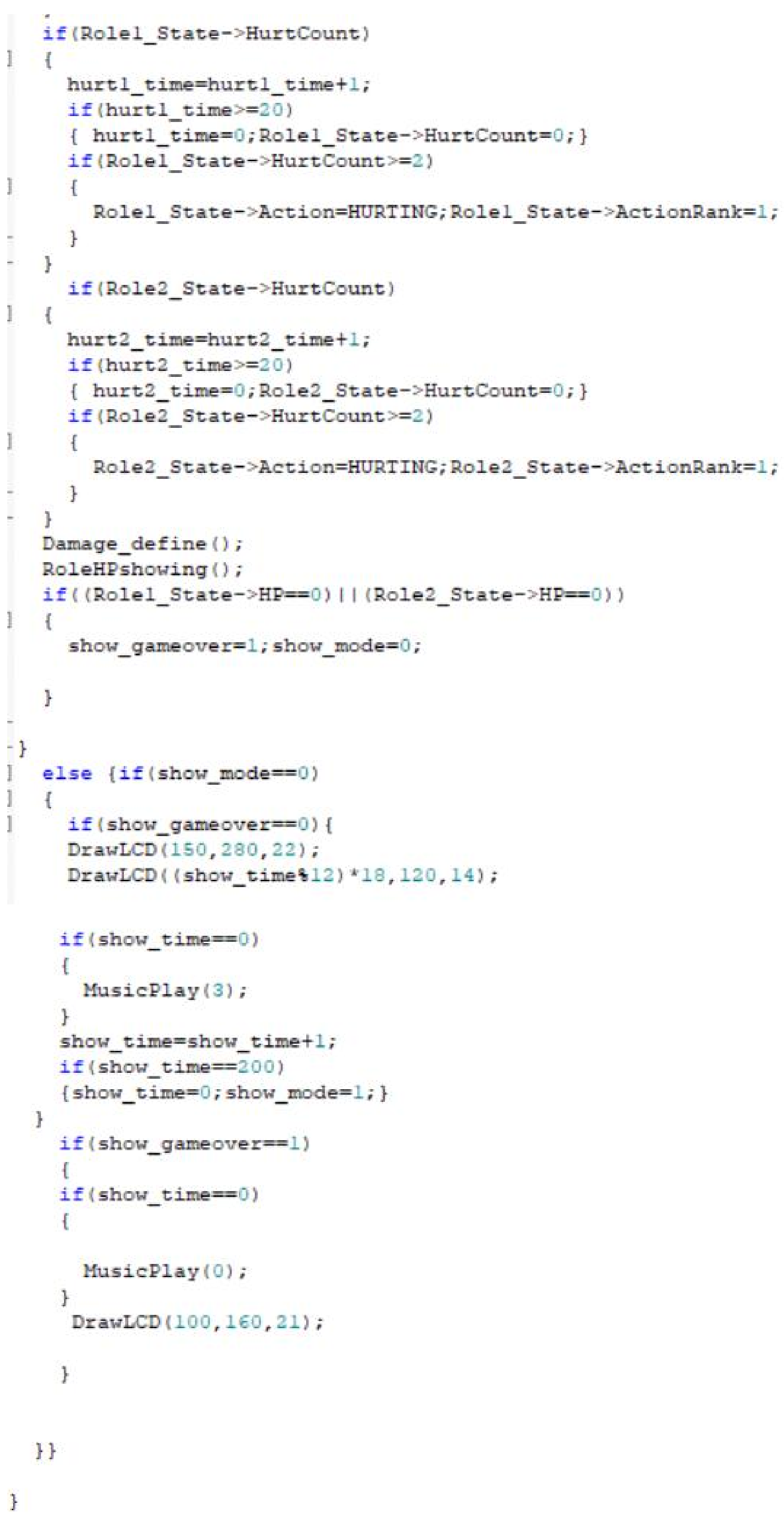
4.2 定时器中断



```
void TIMER(void)//100ms
{
  MapDraw();game_time=game_time+1;
  USER->LED=(0x0001<<(game_time%5))|0x1;
  if(Role1_State->HP>100)
    Role1_State->HP=0;
  if(Role2_State->HP>100)
    Role2_State->HP=0;
  //movement//
  if(show_mode==1)
  {
    if(ready_time_en)
    {
      ready_time=ready_time+1;

    }else{
      ready_time=0;
    }
    if(ready_time2_en)
    {
      ready_time2=ready_time2+1;

    }else{
      ready_time2=0;
    }
```



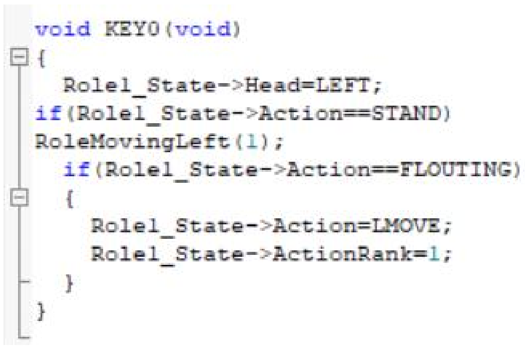
```
switch(Role1_State->Action)
{
  case STAND:Standing1();break;
  case LATTACK:{LeftAttacking1();}break;
  case RATTACK:{RightAttacking1();}break;
  case LMOVE:{LeftMoving1();}break;
  case RMOVE:{RightMoving1();}break;
  case JUMP:{Jumping1();}break;
  case HURTING:{Hurting1();}break;
  case SHOOT:{Shooting1();}break;
  case FLOUTING:{Floating1();}break;
  case DOWN:{Down1();}break;
  case SQUAT:{Squating1();}break;
  default:break;

}
  switch(Map_State->DamageBlock1)
{
  case ZERO:break;
  case FIRE:{Fire();}break;
  case ICE :{Ice();}break;
  default:break;

}
  switch(Map_State->DamageBlock3)
{
  case ZERO:break;
  case FIRE:{FireGo();}break;
  case ICE :{Ice();}break;
  default:break;

}
  switch(Map_State->DamageBlock4)
{
  case ZERO:break;
  case FIRE:{Fire();}break;
  case ICE :{Ice();}break;
  default:break;

}
  switch(Map_State->DamageBlock6)
{
  case ZERO:break;
  case FIRE:break;
  case ICE :{IceGo();}break;
  default:break;

}
  switch(Role2_State->Action)
{
  case STAND:Standing2();break;
  case LATTACK:{LeftAttacking2();}break;
  case RATTACK:{RightAttacking2();}break;
  case LMOVE:{LeftMoving2();}break;
  case RMOVE:{RightMoving2();}break;
  case JUMP:{Jumping2();}break;
  case HURTING:{Hurting2();}break;
  case SHOOT:{Shooting2();}break;
  case FLOUTING:{Floating2();}break;
  case DOWN:{Down2();}break;
  case SQUAT:{Squating2();}break;
  default:break;

}
```



```
 if(Role1_State->HurtCount)
 {
   hurt1_time=hurt1_time+1;
   if(hurt1_time>=20)
   { hurt1_time=0;Role1_State->HurtCount=0;}
   if(Role1_State->HurtCount>=2)
   {
     Role1_State->Action=HURTING;Role1_State->ActionRank=1;
   }
 }
   if(Role2_State->HurtCount)
 {
   hurt2_time=hurt2_time+1;
   if(hurt2_time>=20)
   { hurt2_time=0;Role2_State->HurtCount=0;}
   if(Role2_State->HurtCount>=2)
   {
     Role2_State->Action=HURTING;Role2_State->ActionRank=1;
   }
 }
 Damage_define();
 RoleHPshowing();
 if((Role1_State->HP==0)||(Role2_State->HP==0))
 {
   show_gameover=1;show_mode=0;

 }

}
 else {if(show_mode==0)
 {
   if(show_gameover==0){
   DrawLCD(150,280,22);
   DrawLCD((show_time%12)*18,120,14);

  if(show_time==0)
  {
    MusicPlay(3);
  }
  show_time=show_time+1;
  if(show_time==200)
  {show_time=0;show_mode=1;}
 }
  if(show_gameover==1)
  {
  if(show_time==0)
  {

    MusicPlay(0);
  }
   DrawLCD(100,160,21);

  }


 }}

}
```

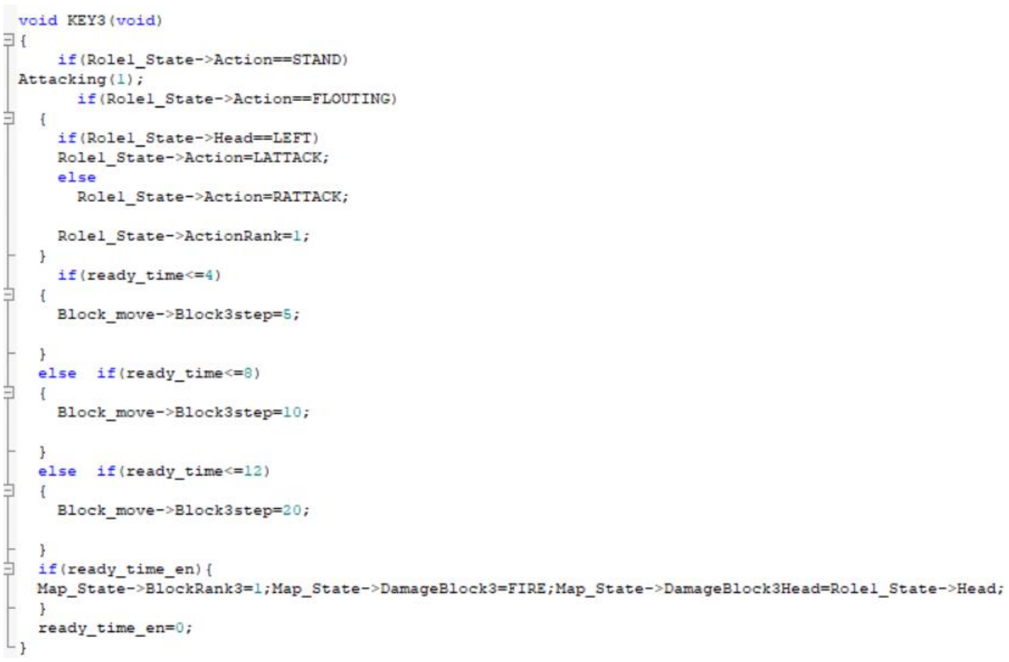
4.3 按键中断函数

示例：KEY0（Role1 左移）



```c
void KEY0(void)
{
   Role1_State->Head=LEFT;
if(Role1_State->Action==STAND)
RoleMovingLeft(1);
   if(Role1_State->Action==FLOUTING)
   {
     Role1_State->Action=LMOVE;
     Role1_State->ActionRank=1;
   }
}
```

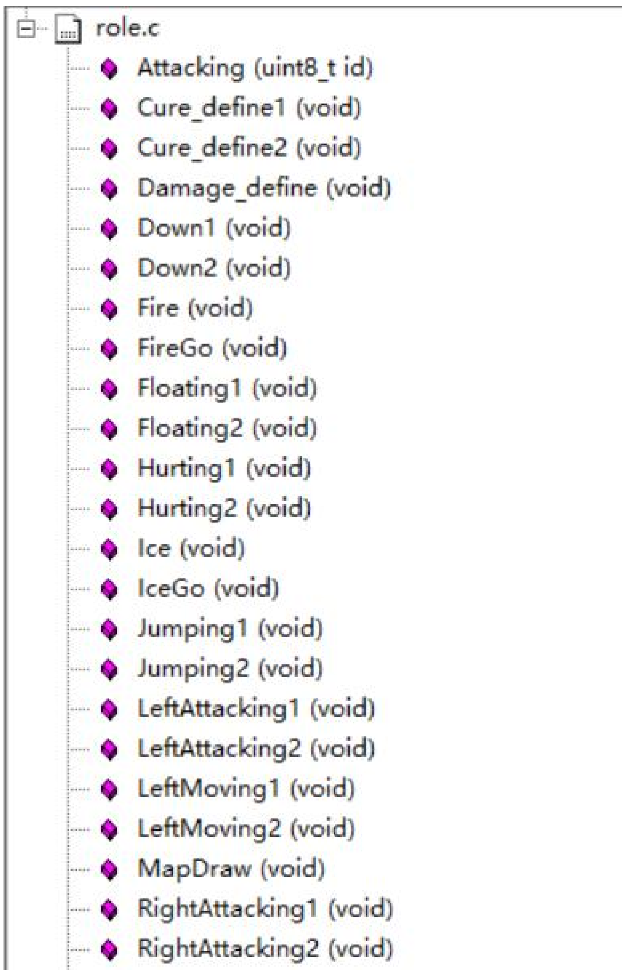
KEY3（Role1 攻击）



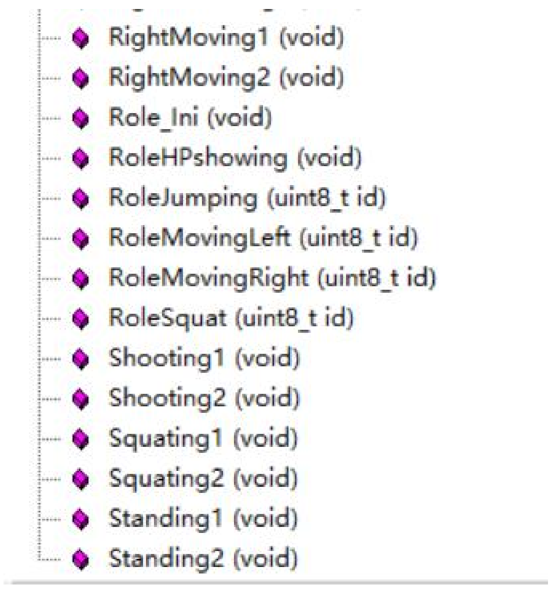
```c
void KEY3(void)
{
    if(Role1_State->Action==STAND)
Attacking(1);
      if(Role1_State->Action==FLOUTING)
  {
    if(Role1_State->Head==LEFT)
    Role1_State->Action=LATTACK;
    else
      Role1_State->Action=RATTACK;

    Role1_State->ActionRank=1;
  }
    if(ready_time<=4)
  {
    Block_move->Block3step=5;

  }
  else  if(ready_time<=8)
  {
    Block_move->Block3step=10;

  }
  else  if(ready_time<=12)
  {
    Block_move->Block3step=20;

  }
  if(ready_time_en){
  Map_State->BlockRank3=1;Map_State->DamageBlock3=FIRE;Map_State->DamageBlock3Head=Role1_State->Head;
  }
  ready_time_en=0;
}
```

4.4 游戏函数

函数目录：



role.c
- Attacking (uint8_t id)
- Cure_define1 (void)
- Cure_define2 (void)
- Damage_define (void)
- Down1 (void)
- Down2 (void)
- Fire (void)
- FireGo (void)
- Floating1 (void)
- Floating2 (void)
- Hurting1 (void)
- Hurting2 (void)
- Ice (void)
- IceGo (void)
- Jumping1 (void)
- Jumping2 (void)
- LeftAttacking1 (void)
- LeftAttacking2 (void)
- LeftMoving1 (void)
- LeftMoving2 (void)
- MapDraw (void)
- RightAttacking1 (void)
- RightAttacking2 (void)

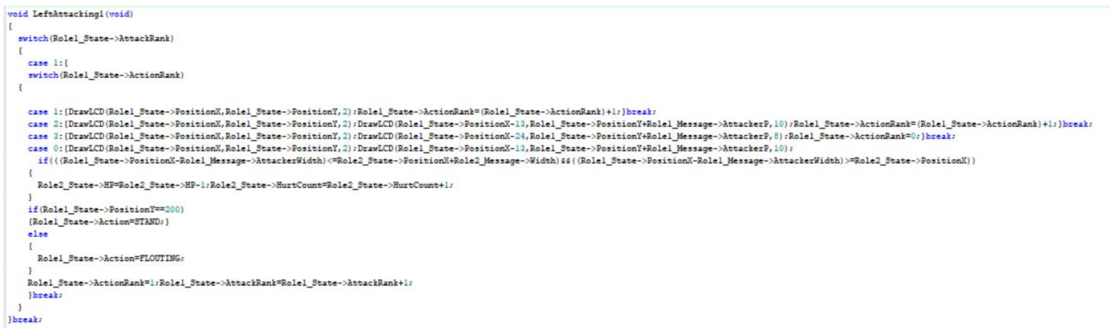


- RightMoving1 (void)
- RightMoving2 (void)
- Role_Ini (void)
- RoleHPshowing (void)
- RoleJumping (uint8_t id)
- RoleMovingLeft (uint8_t id)
- RoleMovingRight (uint8_t id)
- RoleSquat (uint8_t id)
- Shooting1 (void)
- Shooting2 (void)
- Squating1 (void)
- Squating2 (void)
- Standing1 (void)
- Standing2 (void)

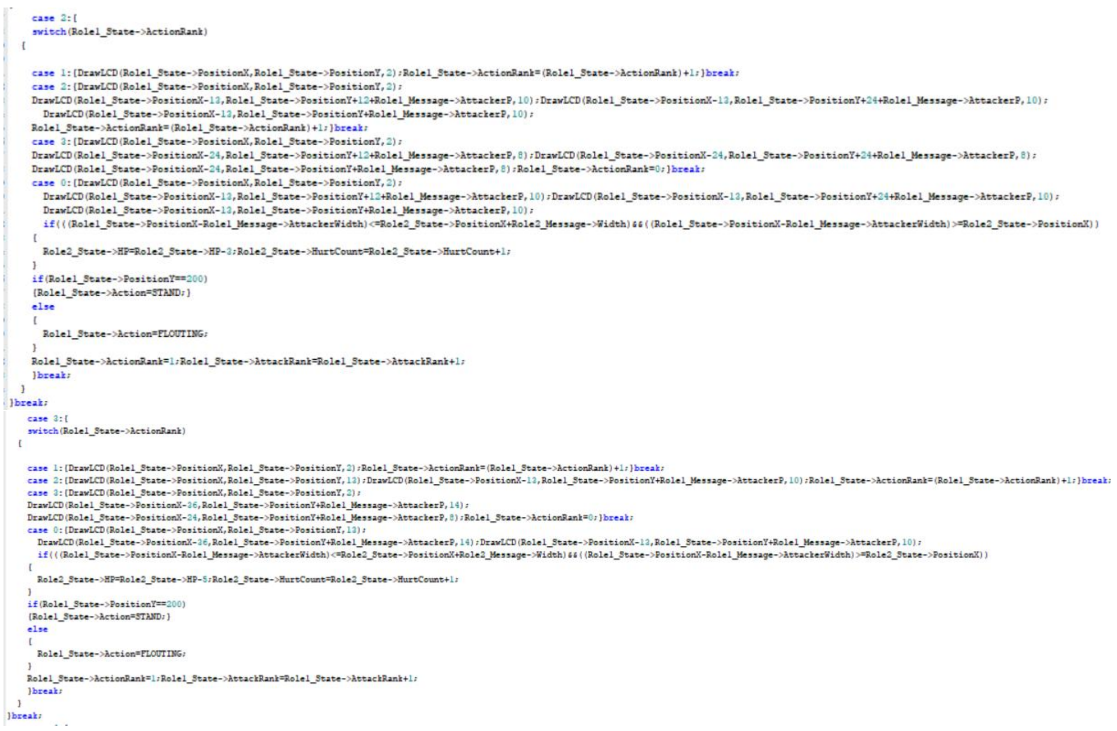
基本数据结构体：

```c
typedef struct{
    volatile uint8_t HP;
    volatile uint8_t PositionX;
    volatile uint16_t PositionY;
    volatile uint8_t Action;
    volatile uint8_t ActionRank;
    volatile uint8_t HurtCount;
    volatile uint8_t Head;
    volatile uint8_t AttackRank;
}role_state;
typedef struct{
    volatile uint8_t Height;
    volatile uint8_t Width;
    volatile uint8_t AttackerP;
    volatile uint8_t AttackerHeight;
    volatile uint8_t AttackerWidth;

}role message;
typedef struct{
    volatile uint8_t Height;
    volatile uint8_t DamageBlock1;
    volatile uint8_t DamageBlock2;
    volatile uint8_t DamageBlock3;
    volatile uint8_t BlockRank1;
    volatile uint8_t BlockRank2;
    volatile uint8_t BlockRank3;
    volatile uint8_t DamageBlock1X;
    volatile uint8_t DamageBlock2X;
    volatile uint8_t DamageBlock3X;
    volatile uint16_t DamageBlock1Y;
    volatile uint16_t DamageBlock2Y;
    volatile uint16_t DamageBlock3Y;
    volatile uint8_t DamageBlock1Head;
    volatile uint8_t DamageBlock2Head;
    volatile uint8_t DamageBlock3Head;
    volatile uint8_t DamageBlock4;
    volatile uint8_t DamageBlock5;
    volatile uint8_t DamageBlock6;
    volatile uint8_t BlockRank4;
    volatile uint8_t BlockRank5;
    volatile uint8_t BlockRank6;
    volatile uint8_t DamageBlock4X;
    volatile uint8_t DamageBlock5X;
    volatile uint8_t DamageBlock6X;
    volatile uint16_t DamageBlock4Y;
    volatile uint16_t DamageBlock5Y;
    volatile uint16_t DamageBlock6Y;
    volatile uint8_t DamageBlock4Head;
    volatile uint8_t DamageBlock5Head;
    volatile uint8_t DamageBlock6Head;
}map_state;

typedef struct{
    volatile uint8_t Block1step;
    volatile uint8_t Block2step;
    volatile uint8_t Block3step;
    volatile uint8_t Block4step;
    volatile uint8_t Block5step;
    volatile uint8_t Block6step;
}block_move;
```

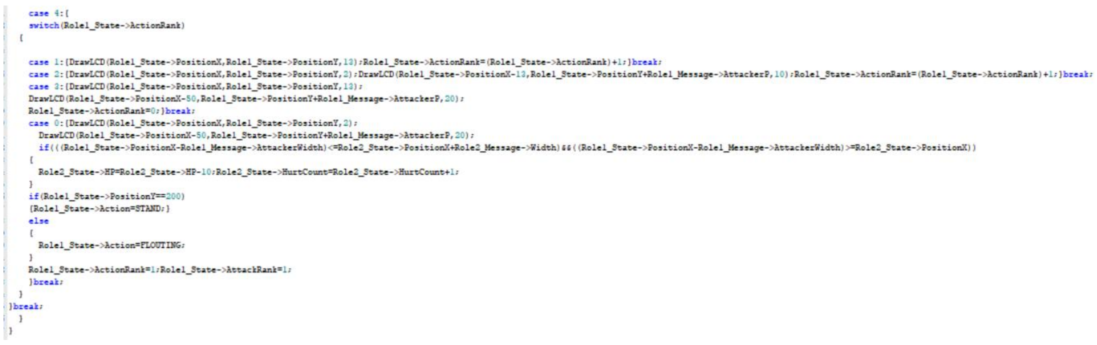
动作处理函数示例：Role1 左攻击



```c
void LeftAttacking1(void)
{
  switch(Role1_State->AttackRank)
  {
    case 1:{
    switch(Role1_State->ActionRank)
  {

    case 1:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,2);Role1_State->ActionRank=(Role1_State->ActionRank)+1;}break;
    case 2:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,2);DrawLCD(Role1_State->PositionX-13,Role1_State->PositionY+Role1_Message->AttackerP,10);Role1_State->ActionRank=(Role1_State->ActionRank)+1;}break;
    case 3:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,2);DrawLCD(Role1_State->PositionX-24,Role1_State->PositionY+Role1_Message->AttackerP,8);Role1_State->ActionRank=0;}break;
    case 0:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,2);DrawLCD(Role1_State->PositionX-13,Role1_State->PositionY+Role1_Message->AttackerP,10);
      if(((Role1_State->PositionX-Role1_Message->AttackerWidth)<=Role2_State->PositionX+Role2_Message->Width)&&((Role1_State->PositionX-Role1_Message->AttackerWidth)>=Role2_State->PositionX))
     {
      Role2_State->HP=Role2_State->HP-1;Role2_State->HurtCount=Role2_State->HurtCount+1;
     }
    if(Role1_State->PositionY==200)
    {Role1_State->Action=STAND;}
    else
     {
      Role1_State->Action=FLOUTING;
     }
    Role1_State->ActionRank=1;Role1_State->AttackRank=Role1_State->AttackRank+1;
    }break;
  }
}break;
```

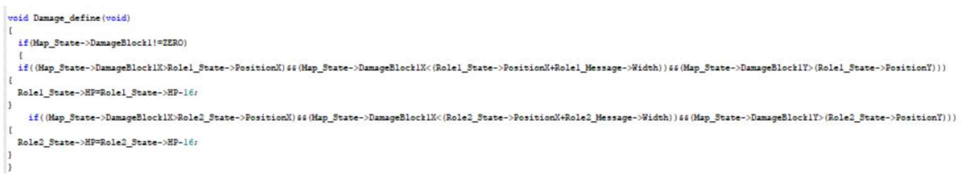


```c
     case 2:{
     switch(Role1_State->ActionRank)
   {

     case 1:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,2);Role1_State->ActionRank=(Role1_State->ActionRank)+1;}break;
     case 2:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,2);
     DrawLCD(Role1_State->PositionX-13,Role1_State->PositionY+12+Role1_Message->AttackerP,10);DrawLCD(Role1_State->PositionX-13,Role1_State->PositionY+24+Role1_Message->AttackerP,10);
       DrawLCD(Role1_State->PositionX-13,Role1_State->PositionY+Role1_Message->AttackerP,10);
     Role1_State->ActionRank=(Role1_State->ActionRank)+1;}break;
     case 3:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,2);
     DrawLCD(Role1_State->PositionX-24,Role1_State->PositionY+12+Role1_Message->AttackerP,8);DrawLCD(Role1_State->PositionX-24,Role1_State->PositionY+24+Role1_Message->AttackerP,8);
     DrawLCD(Role1_State->PositionX-24,Role1_State->PositionY+Role1_Message->AttackerP,8);Role1_State->ActionRank=0;}break;
     case 0:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,2);
       DrawLCD(Role1_State->PositionX-13,Role1_State->PositionY+12+Role1_Message->AttackerP,10);DrawLCD(Role1_State->PositionX-13,Role1_State->PositionY+24+Role1_Message->AttackerP,10);
       DrawLCD(Role1_State->PositionX-13,Role1_State->PositionY+Role1_Message->AttackerP,10);
       if(((Role1_State->PositionX-Role1_Message->AttackerWidth)<=Role2_State->PositionX+Role2_Message->Width)&&((Role1_State->PositionX-Role1_Message->AttackerWidth)>=Role2_State->PositionX))
      {
       Role2_State->HP=Role2_State->HP-3;Role2_State->HurtCount=Role2_State->HurtCount+1;
      }
     if(Role1_State->PositionY==200)
     {Role1_State->Action=STAND;}
     else
      {
       Role1_State->Action=FLOUTING;
      }
     Role1_State->ActionRank=1;Role1_State->AttackRank=Role1_State->AttackRank+1;
     }break;
   }
}break;
    case 3:{
    switch(Role1_State->ActionRank)
  {

    case 1:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,2);Role1_State->ActionRank=(Role1_State->ActionRank)+1;}break;
    case 2:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,13);DrawLCD(Role1_State->PositionX-13,Role1_State->PositionY+Role1_Message->AttackerP,10);Role1_State->ActionRank=(Role1_State->ActionRank)+1;}break;
    case 3:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,2);
    DrawLCD(Role1_State->PositionX-36,Role1_State->PositionY+Role1_Message->AttackerP,14);
    DrawLCD(Role1_State->PositionX-24,Role1_State->PositionY+Role1_Message->AttackerP,8);Role1_State->ActionRank=0;}break;
    case 0:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,13);
      DrawLCD(Role1_State->PositionX-36,Role1_State->PositionY+Role1_Message->AttackerP,14);DrawLCD(Role1_State->PositionX-13,Role1_State->PositionY+Role1_Message->AttackerP,10);
      if(((Role1_State->PositionX-Role1_Message->AttackerWidth)<=Role2_State->PositionX+Role2_Message->Width)&&((Role1_State->PositionX-Role1_Message->AttackerWidth)>=Role2_State->PositionX))
     {
      Role2_State->HP=Role2_State->HP-5;Role2_State->HurtCount=Role2_State->HurtCount+1;
     }
    if(Role1_State->PositionY==200)
    {Role1_State->Action=STAND;}
    else
     {
      Role1_State->Action=FLOUTING;
     }
    Role1_State->ActionRank=1;Role1_State->AttackRank=Role1_State->AttackRank+1;
    }break;
  }
}break;
```



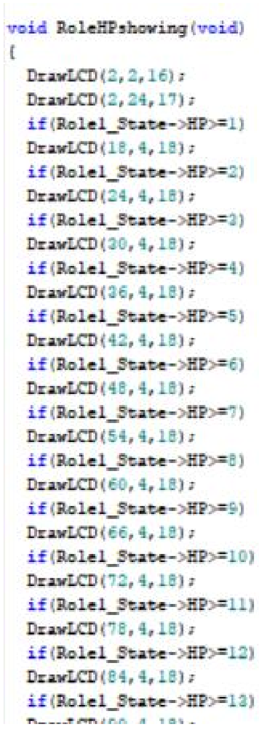
```c
    case 4:{
    switch(Role1_State->ActionRank)
  {

    case 1:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,13);Role1_State->ActionRank=(Role1_State->ActionRank)+1;}break;
    case 2:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,2);DrawLCD(Role1_State->PositionX-13,Role1_State->PositionY+Role1_Message->AttackerP,10);Role1_State->ActionRank=(Role1_State->ActionRank)+1;}break;
    case 3:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,13);
    DrawLCD(Role1_State->PositionX-50,Role1_State->PositionY+Role1_Message->AttackerP,20);
    Role1_State->ActionRank=0;}break;
    case 0:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,2);
      DrawLCD(Role1_State->PositionX-50,Role1_State->PositionY+Role1_Message->AttackerP,20);
      if(((Role1_State->PositionX-Role1_Message->AttackerWidth)<=Role2_State->PositionX+Role2_Message->Width)&&((Role1_State->PositionX-Role1_Message->AttackerWidth)>=Role2_State->PositionX))
     {
      Role2_State->HP=Role2_State->HP-10;Role2_State->HurtCount=Role2_State->HurtCount+1;
     }
    if(Role1_State->PositionY==200)
    {Role1_State->Action=STAND;}
    else
     {
      Role1_State->Action=FLOUTING;
     }
    Role1_State->ActionRank=1;Role1_State->AttackRank=1;
    }break;
  }
}break;
  }
}
```

法球伤害判定函数示例：



```c
void Damage_define(void)
{
  if(Map_State->DamageBlock1!=ZERO)
  {
  if((Map_State->DamageBlock1X>Role1_State->PositionX)&&(Map_State->DamageBlock1X<(Role1_State->PositionX+Role1_Message->Width))&&(Map_State->DamageBlock1Y>(Role1_State->PositionY)))
{
  Role1_State->HP=Role1_State->HP-16;
}
    if((Map_State->DamageBlock1X>Role2_State->PositionX)&&(Map_State->DamageBlock1X<(Role2_State->PositionX+Role2_Message->Width))&&(Map_State->DamageBlock1Y>(Role2_State->PositionY)))
{
  Role2_State->HP=Role2_State->HP-16;
}
}
```

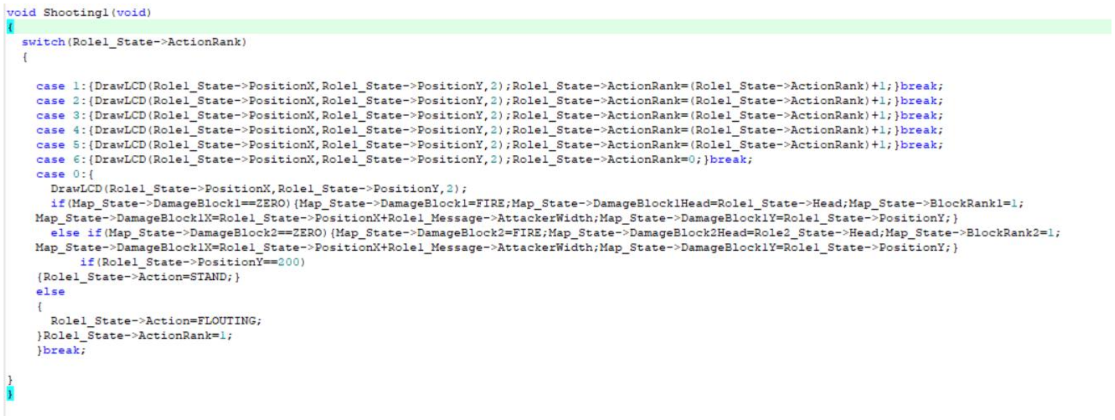
血量显示函数:



```c
void RoleHPshowing(void)
{
  DrawLCD(2,2,16);
  DrawLCD(2,24,17);
  if(Role1_State->HP>=1)
  DrawLCD(18,4,18);
  if(Role1_State->HP>=2)
  DrawLCD(24,4,18);
  if(Role1_State->HP>=3)
  DrawLCD(30,4,18);
  if(Role1_State->HP>=4)
  DrawLCD(36,4,18);
  if(Role1_State->HP>=5)
  DrawLCD(42,4,18);
  if(Role1_State->HP>=6)
  DrawLCD(48,4,18);
  if(Role1_State->HP>=7)
  DrawLCD(54,4,18);
  if(Role1_State->HP>=8)
  DrawLCD(60,4,18);
  if(Role1_State->HP>=9)
  DrawLCD(66,4,18);
  if(Role1_State->HP>=10)
  DrawLCD(72,4,18);
  if(Role1_State->HP>=11)
  DrawLCD(78,4,18);
  if(Role1_State->HP>=12)
  DrawLCD(84,4,18);
  if(Role1_State->HP>=13)
```

......

火球召唤函数:



```c
void Shooting1(void)
{
  switch(Role1_State->ActionRank)
  {

    case 1:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,2);Role1_State->ActionRank=(Role1_State->ActionRank)+1;}break;
    case 2:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,2);Role1_State->ActionRank=(Role1_State->ActionRank)+1;}break;
    case 3:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,2);Role1_State->ActionRank=(Role1_State->ActionRank)+1;}break;
    case 4:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,2);Role1_State->ActionRank=(Role1_State->ActionRank)+1;}break;
    case 5:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,2);Role1_State->ActionRank=(Role1_State->ActionRank)+1;}break;
    case 6:{DrawLCD(Role1_State->PositionX,Role1_State->PositionY,2);Role1_State->ActionRank=0;}break;
    case 0:{
      DrawLCD(Role1_State->PositionX,Role1_State->PositionY,2);
      if(Map_State->DamageBlock1==ZERO){Map_State->DamageBlock1=FIRE;Map_State->DamageBlock1Head=Role1_State->Head;Map_State->BlockRank1=1;
    Map_State->DamageBlock1X=Role1_State->PositionX+Role1_Message->AttackerWidth;Map_State->DamageBlock1Y=Role1_State->PositionY;}
      else if(Map_State->DamageBlock2==ZERO){Map_State->DamageBlock2=FIRE;Map_State->DamageBlock2Head=Role2_State->Head;Map_State->BlockRank2=1;
    Map_State->DamageBlock1X=Role1_State->PositionX+Role1_Message->AttackerWidth;Map_State->DamageBlock1Y=Role1_State->PositionY;}
          if(Role1_State->PositionY==200)
    {Role1_State->Action=STAND;}
    else
    {
      Role1_State->Action=FLOUTING;
    }Role1_State->ActionRank=1;
    }break;

}
}
```