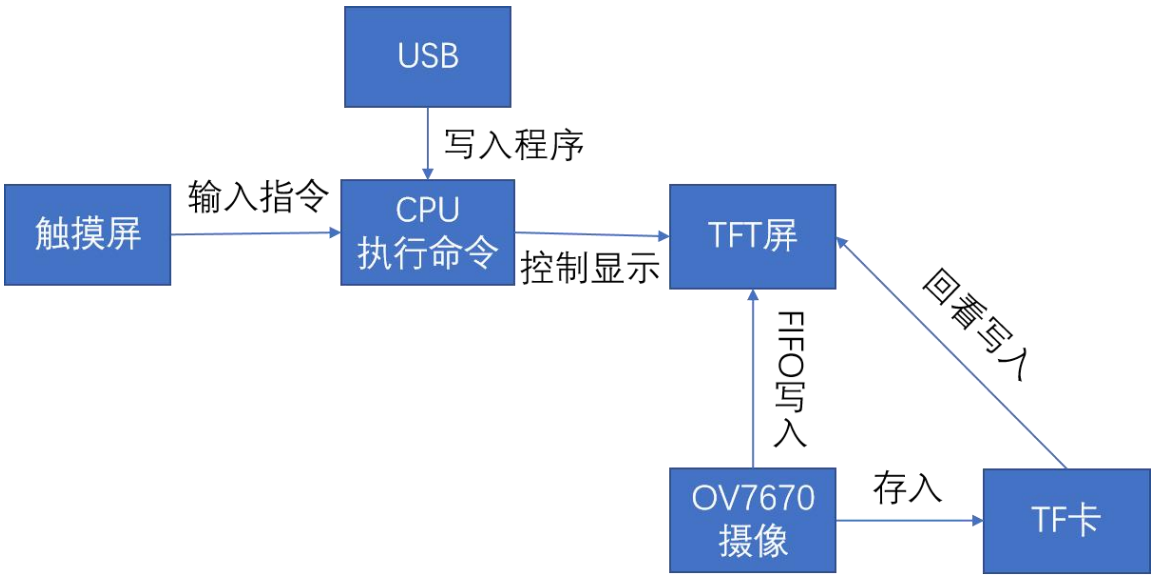
# 一、实验要求

在掌握 C8051 单片机应用系统设计方法和 C51 编程的基础上，以基于 IAP15W4K61S4 单片机开发板为相机控制器平台，研究 OV7670 CMOS 模组接口设计、TFT LCD 和 4 线电阻触摸屏接口设计、基于 SD 存储卡的图像文件存取设计等基本应用程序的开发方法，完成一个 30 万像元的数码相机原理样机的集成功能演示，可以实现对摄像头采集的 240×240 彩色图像在 TFT LCD 上的实时显示，并能以 BMP 格式存储在 SD 卡上，同时设计友好的 GUI 用户操作界面，能够实现相机的图像采集、存储和回放功能。

# 二、硬件系统设计

（要给出**原理图**、主要硬件（CPU、RAM、SD 卡、摄像头、液晶、触摸屏等）的**参数**）

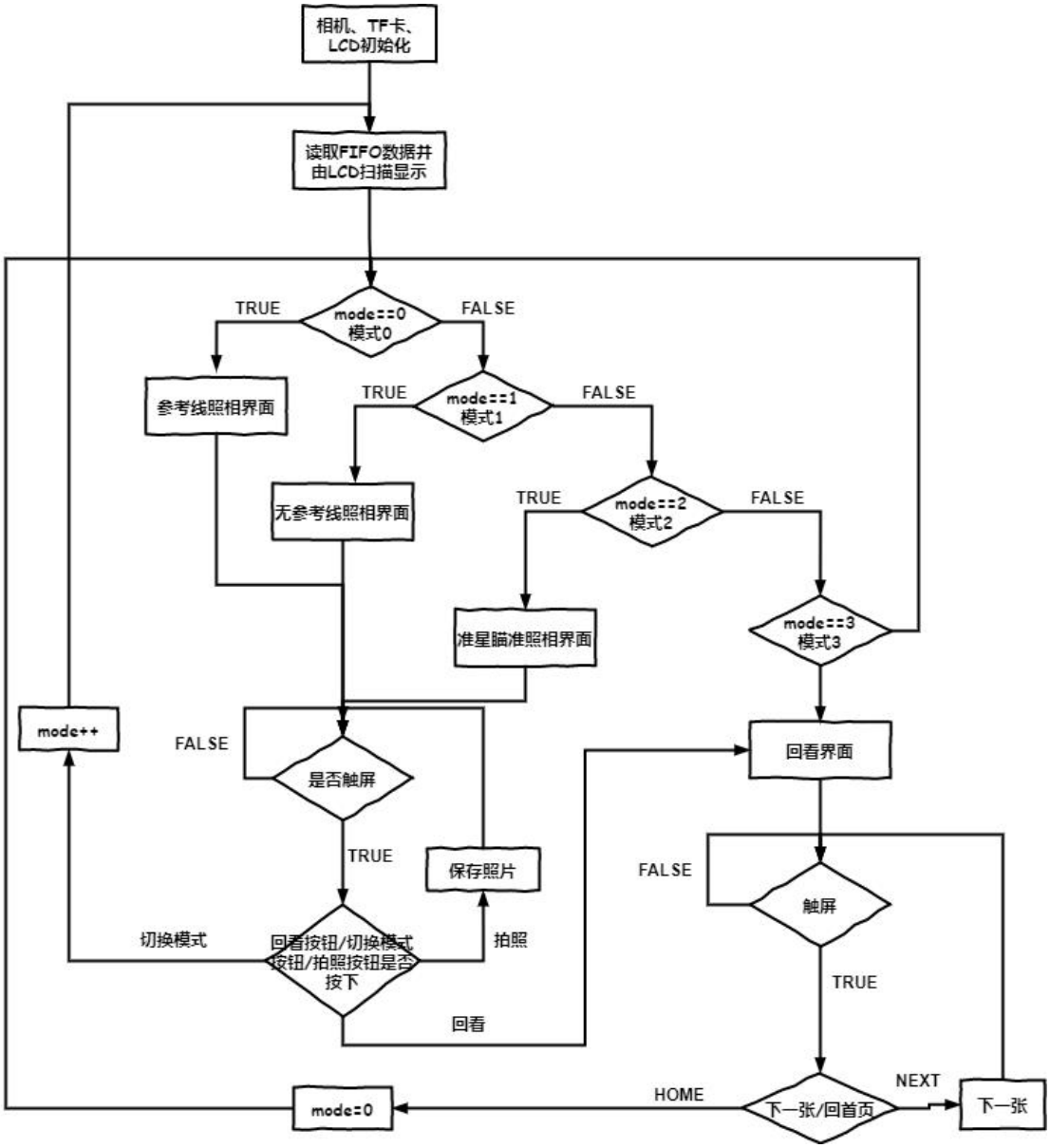



硬件参数

| 硬件 | 规格参数 | 性能参数 |
|---|---|---|
| CPU | 增强型 8051 | 单时钟/机器周期 |
| RAM | 1024KB | 兼容传统 8051 |
| SD 卡（TF 卡） | 24mm ×32mm ×1.44mm | 128MB |
| 摄像头 | 2.36mmx1.76mm | 30 帧/秒 |
| 触摸屏 | 240*320 | RGB |

# 三、程序实现框图

（要给出 main.c 和自己定义的.c 或者.asm 文件的流程图）




# 四、程序源码

（要给出 main.c 和自己定义的.c 文件、.h 文件及.asm 文件的源码及**注释**）

1.main.c 文件

```
#include   "stc15f2k60s2.h"        //STC15 头文件
#include   "def.h"                 //宏定义  常用函数
#include   "delay.h"               //延时函数
```

```
#include    "tft.h"                      //TFT IC 底层驱动
#include    "gui.h"
#include    "xpt2046.h"                  //触摸屏驱动
#include    "spi.h"
#include    "key.h"
#include    "pff.h"                      //文件系统调用.h 加载
#include    "sram.h"
#include    "flash.h"
#include    "sd.h"
#include    "ov7670.h"
#include <string.h>



FATFS fatfs;            //文件系统结构体定义
u8 tbuf[512];
u8       pic_width;   //图片宽度
u16 pic_heigh ;       //图片高度
u8 mode = 0;          //工作模式，0 有参考线，1 无参考线，2 瞄准线，3 回看
//针对  IAP15W4K61S4  系列  IO 口初始化
//io 口初始化  P0 P1 P2 P3 P4  为准双向 IO 口
//注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
//          高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
//相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
//             P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
void IO_init(void)
{
   P0M0 = 0X00;
   P0M1 = 0X00;
   P1M0 = 0X00;
   P1M1 = 0X00;
   P2M0 = 0X00;
   P2M1 = 0X00;
   P3M0 = 0X00;
   P3M1 = 0X00;
   P4M0 = 0X00;
   P4M1 = 0X00;
}
//将屏幕从（0,0）位置到(240,240)位置  截取 240x240 图片数据  并转换为 BMP 格式存入 TF 卡中
//说明：  默认的文件名字"/OV76/M1.bmp"  别的名字也可以，但必须得在卡上有这个文件存在
//     写入是破坏式写入
//函数思路：因为 petit fatfs 文件系统只能在原有的文件上更新，而 BMP 图片前 54 字节
// 是图片的信息  这一部分是不要更改  保存原有就行
// 所以程序上要先将前 54 字节取出  然后后在更新 54 字节以后的图片数据
```

```
//注意：petit fatfs 文件系统 在写数据时必须要从 扇区的开头写起 不弄成扇区的中间写数据
// 所以在函数提取了前 54 字节后 在和后面的颜色数据组成 512 字节 在重新写回扇区
//*path 保存路径
//返回 0 保存成功 1 保存失败
u8 Show_Bmp(const char *path)
{
    FRESULT res;
    u16 br,y=0,zy,height,    //width,height 图片的初始左边
    y1,i1,tmp;                          //tmp 16 位解码变量
    u8 x=0,zx,width,x1,
    rgb=0, Bmpcolor;
    res=pf_open(path);        //打开指定路径文件 这一步可以打开任何路径下的文件 注意它的功能就是打开文件，不是文件夹
                              //读文件内数据的前提是必须要打开这个文件
    if(res == FR_OK)
    {
        pf_read(tbuf, 54, &br);          //取前 54 字节 前 54 字节含有 bmp 文件大小 文件长宽尺度 像素值 等数据信息
        if(br!=54) return 1;          //提取出错
    //实际宽和高都是用 4 个字节表示的，但图片的大小不能超过屏的尺寸
    //故这里只用一个字节表示宽度,两个字节表示高度
        width  = tbuf[18];                          //计算图片宽度
        height = (tbuf[23]<<8)+tbuf[22];  //计算图片高度
        pic_width  = width;
        pic_height = height;
        Bmpcolor=tbuf[28]/8;    //获取颜色深度 一般是 16 位 24 位 32 位
            //number(30,280,Bmpcolor,Red,White);
            //将小于屏幕尺寸的图片放到屏幕正中间 顶头显示
        if(width<239)    zx=(240-width)/2;          else zx=0;
        if(height<299)   zy=(320-height);           else zy=0;
        x1=zx; y1=zy;                  //赋值计算后的值
        LCD_scan(2);      //BMP 图片解码的扫描方式为 从左到右    从下到上 否则显示的图片上下颠倒
        Address_set(x1,y1,x1+width-1,y1+height-1); //设置显示范围 先扫横行 再扫纵行
        LCD_RS=1;     //写数据线拉高    为提高写入速度 主循环前拉高
        while(1)                       //一直到最后一簇
        {
            pf_read(tbuf, 512, &br);//从 54 字节后位置读取 512 个字节到缓存区
            for(i1=0;i1<512;i1++)
            {
                if(Bmpcolor==2)              //16 位 BMP
                {
```

```
                switch(rgb)                    //555 转 565 格式
                {
                    case 0:
                        tmp=((u16)tbuf[i1]&0x1f);                //R
                        tmp+=(((u16)tbuf[i1])&0xe0)<<1;  //G
                        break;
                    case 1:
                        tmp+=(u16)tbuf[i1]<<9;                   //B
                        break;
                }
            }
            else if(Bmpcolor==3)//24 位 BMP 解码  RGB 分别占 8 个字节
            {
                switch(rgb)
                {
                    case 0:
                        tmp=tbuf[i1]>>3;                         //B
                        break;
                    case 1:
                        tmp+=((u16)tbuf[i1]<<3)&0x07e0;          //G
                        break;
                    case 2:
                        tmp+=((u16)tbuf[i1]<<8)&0xf800;          //R
                        break;
                }
            }
            rgb++;
            if(rgb==Bmpcolor)
            {
                P2=tmp>>8; //为了提高显示速度  直接调用 IO 口本身
                P0=tmp;      //void Lcd_Write_Data(u16 Data)函数的分解
                tmp=0;
                rgb=0;
                LCD_WR=0;         //开始写入
                LCD_WR=1;
                x1++;//横向自加  加满一行  横向清零  纵向自加  直到扫
完整体图片
                if(x1==width+zx)
                {
                    y1--;
                    x1=zx;
                    if(y1==zy-height)
                    {                //恢复正常扫描方式
                        LCD_scan(1);
                        return 0;       //显示完成
                    }
                }
            }
        }
```

```
            }
        }
        return 1;    //出错
    }
    //lcd 屏幕扫描，通过设置分辨率，将 FIFO 的图像数据加载到 LCD 之中
    //图片显示的过程中属于中断关闭，无法接收触摸屏产生的外部中断信号
    void scan(void)
    {
        u32 j;
        Address_set(0,0,239,239);    //设置显示范围   显示为 240*240
        if(cur_status==2)             //判断缓存区是否存好摄像数据
        {
            FIFO_RRST=0;              //开始复位读指针     读取 FIFO 数据
            FIFO_RCLK=0;
            FIFO_RCLK=1;
            FIFO_RCLK=0;
            FIFO_RRST=1;              //复位读指针结束
            FIFO_RCLK=1;
            LCD_RS=1;                 //拉高 TFT  写数据端
            for(j =0;j<57600;j++)     //分辨率为 240x240     每个颜色点要两个字节  所以  240x240x2=57600    次
            {
                FIFO_RCLK=0;          //每一次时钟跳变  读一次数据
                P2=P1;                //直接将高字节数据给 P2
                FIFO_RCLK=1;
                FIFO_RCLK=0;
                P0=P1;                    //直接将低字节数据给 P0
                FIFO_RCLK=1;
                LCD_WR=0;                 //  显示
                LCD_WR=1;
            }
            EX0 = 1;         //显示完成开中断
            cur_status=0; //显示完整个图片以后将 cur_status 置 0 准备接收下一帧
        }
    }
    //保存图片函数，将 LCD 显示的图片的 240*240 的部分存于外部数据存储器
    //保存图片时 LED 亮起表示正常储存
    u8 Save_Bmp(const char *path)
    {
        FRESULT res;
        u16 br,i,j,m=0,n=239,color;
        sram(1);                          //开启外部内存
    //由于文件路径在外部 SRAM 中  所以这里要开启外部 SRAM 才能调用路径
        res=pf_open(path);                        //打开指定路径文件     这一步可以打开任何路径下的文件  注意它的功能就是打开文件，不是文件夹
                                  //读文件内数据的前提是必须要打开这个文件
        if(res == FR_OK)
```

```
        {
            led = 0; //存盘指示灯(P3.5)亮
            pf_read(tbuf,54,&br);      //提取 BMP 图片前 54 字节图片信息
            pf_open(path);   //重新打开路径 将扇区指向图片首数据位置
            sram(0);                          //关闭外部内存，开启液晶片选
            for(i=27;i<256;i++)        //提取 512 个字节到 tbuf 中 即 256 个颜色点
            {
                color=LCD_readpoint(m,n);   //提取 LCD 每个像元的颜色
                color=((color>>1)&0x7fe0)+(color&0x001f);   //将提取的 565 格式
转换为 555 格式
                tbuf[i*2]=color;                        //存入时低字节在前
                tbuf[i*2+1]=(color>>8);
                m++;
            }
            pf_write(tbuf,512,&br);         //向 TF 卡内写入 512 个字节（1 个扇
          for(j=0;j<254;j++)
          {
                for(i=0;i<256;i++)          //提取 512 个字节到 tbuf 中 即 256 个颜
色点
                {
                    color=LCD_readpoint(m,n);  //符合摄像头摄像效果提取颜色
                    color=((color>>1)&0x7fe0)+(color&0x001f); //将提取的 565
格式转换为 555 格式
                    tbuf[i*2]=color;                        //存入时低字节在前
                    tbuf[i*2+1]=(color>>8);
                    m++;
                    if(m==240)
                    {
                        m=0;
                        n--;       //这里不用判断 m 因为循环固定 直接会跳出
                    }
                }
              pf_write(tbuf,512,&br);              //向 TF 卡内写入 512 个字节
          }
          SD_DisSelect();//取消 TF 卡片选  在写入函数里加取消片选 会有影响
所以在最后写入完成加取消片选
          delay1ms(2000);       //保持显示延时
          led = 1; //存盘指示灯(P3.5)灭
          return 0;   //写入成功
       }
       else
       {
         return 1;       //错误
       }
    }
    //
    //触摸数据转换屏实际坐标函数体
```

```
//返回  result.x 坐标  result.y 坐标  result.flag=1 表示有键按下
//
/*  触摸屏计算公式 lcdx=xa*tpx+xb;lcdy=ya*tpy+yb;
  lcdx,lcdy 为屏坐标    tpx,tpy 为触屏板值  xa,ya 为比例因子    xb,yb 为偏移量
    计算方法  在屏幕上指定 lcdx,lcdy 位置画出十字图形  分别画在屏幕上的 4 个角位置  用触摸笔分别点击  得到其中的触摸值  根据上面的公式  计算  xa,ya xb,yb    这样就能使得触摸板和屏幕校准*/
    //无校准功能
// result.x=0.065894*a-16;//将得到的 AD 值带入公式  计算 lcd 屏的 x y 坐标
//   result.y=0.084031*b-14;
//加了校准功能    sp->x = ((float)T_i2c.xi/10000)*a+T_i2c.a;
//将得到的 AD 值带入公式  计算 lcd 屏的 x y 坐标
//sp->y = ((float)T_i2c.yi/10000)*b+T_i2c.b;sp->flag = flag;
void GetTouchScreenPos(struct TFT_Pointer * sp)
{
    u16 a,b, flag;                    //临时变量
    u8    ax[8];
    u16 x1,y1;
    u16 x2,y2;

    u8 i2t[8];                    //读 24c02 中  触摸校准参数  临时转换调用数
    //触摸屏校准参数
    struct T_i T_i2c=
    {
        656,
        883,
        -13,
        -12,
    };
        //读取 AD 值并转换为 X Y 坐标
   #define ERR_RANGE 5 //误差范围
    if(AD7843_isClick==0)
    {
        delay1ms(1);
        if(AD7843_isClick==0)
        {
            LCD_CS=1; //xpt 的片选线在 tft 上  防止触摸工作时影响 tft    这里关掉 TFT 使能
            AD7843_CS=0;            //开片选
            SPI_Speed(2);//降低  SPI 通讯速度  使触摸芯片放回数据更稳定
/*这里采用 16 时钟周期采集    因为  此触摸功能采用的是 SPI 总线
而 SPI 功能是只能 8 位传输    XPT2046 的 AD 分辨率为 12 位    需要读两次
根据 XPT2046 手册中  16 时钟周期  时序图  可以看出
发送采集数据    接收一次 SPI 数据后  在发送空功能的 SPI 数据    就会把剩下的部分接收到这样先接收的  是低字节数据    第二次接收的是高字节数据    移位后便是 12 位的 AD 值      */
            ax[0]=SPI_SendByte(0x90);   //送控制字 10010000 即用差分方式
```

读 X 坐标，舍弃读取的数据
ax[0]=SPI_SendByte(0x00); //发送任意数据（最高位不能为 1，和 2046 命令冲突），接收 X 高字节
ax[1]=SPI_SendByte(0xD0); //送控制字 11010000 即用差分方式读 Y 坐标，接收 X 低字节
ax[2]=SPI_SendByte(0x00); //发送任意数据（同上），接收 Y 高字节
ax[3]=SPI_SendByte(0x90); //送控制字 10010000 （第二次）读 X 坐标，接收 Y 低字节
ax[4]=SPI_SendByte(0x00); //发送任意数据（同上），接收 X 高字节
ax[5]=SPI_SendByte(0xD0); //送控制字 11010000 （第二次）读 Y 坐标，接收 X 低字节
ax[6]=SPI_SendByte(0x00); //发送任意数据（同上)，接收 Y 高字节
ax[7]=SPI_SendByte(0x90); //送控制字 10010000 （第三次）读 X 坐标，接收 Y 低字节

```
//提取两次采集值
y1=(ax[0]<<5)|(ax[1]>>3);
y2=(ax[4]<<5)|(ax[5]>>3);
x1=(ax[2]<<5)|(ax[3]>>3);
x2=(ax[6]<<5)|(ax[7]>>3);

if(((x2<=x1&&x1<x2+ERR_RANGE)||(x1<=x2&&x2<x1+ERR_RANGE))//前后两次采样在+-ERR_RANGE 内
&&((y2<=y1&&y1<y2+ERR_RANGE)||(y1<=y2&&y2<y1+ERR_RANGE)))
{
    flag=1;          //打开标志位
    a=(x1+x2)/2;
    b=(y1+y2)/2;
}
else flag=0;
SPI_Speed(0);        //调整 SPI 速度为最高
AD7843_CS=1;         //关片选
LCD_CS=0;
```

```
    }
  }
}
//主函数
void main()
{
    char* fname[30]=          //用一个数组储存 30 张图片的名称
   {"/OV76/M1.bmp",       //保存图片时直接修改名称就可以修改储存路径
     "/OV76/M2.bmp",
    "/OV76/M3.bmp",
```

```
 "/OV76/M4.bmp",
 "/OV76/M5.bmp",
 "/OV76/M6.bmp",
 "/OV76/M7.bmp",
 "/OV76/M8.bmp",
 "/OV76/M9.bmp",
 "/OV76/M10.bmp",
 "/OV76/M11.bmp",
 "/OV76/M12.bmp",
 "/OV76/M13.bmp",
 "/OV76/M14.bmp",
 "/OV76/M15.bmp",
 "/OV76/M16.bmp",
 "/OV76/M17.bmp",
 "/OV76/M18.bmp",
 "/OV76/M19.bmp",
 "/OV76/M20.bmp",
 "/OV76/M21.bmp",
 "/OV76/M22.bmp",
 "/OV76/M23.bmp",
 "/OV76/M24.bmp",
 "/OV76/M25.bmp",
 "/OV76/M26.bmp",
 "/OV76/M27.bmp",
 "/OV76/M28.bmp",
 "/OV76/M29.bmp",
 "/OV76/M30.bmp"};
 int x=0;
 int current = 1;
 int y =300;
 u32   j;                                    //临时变量
 u8   sign=0;                               //初始标志
 u8 cankao=1;                             //参考线开关标志
 struct TFT_Pointer sp;   //定义触摸变量结构体
 struct TFT_Pointer tp; //用来检查触摸屏是否有键按下
 /*   初始化阶段  */
SP=0X80;                                    //  调整堆栈指向     手册 286 页  详解
IO_init();                                 //  真对  IAP15W4K61S4   IO 口初始化
Lcd_Init();                              // TFT 液晶初始化
Init_SPI();                               // SPI 初始化
GUI_Clear(Black);                        //  白色清屏
SD_Init();                               // SD 卡初始化
KEY_Init();
pf_mount(&fatfs);     //初始化 petit FATFS 文件系统   并提取 tf 卡相应数据
      //（这句非常重要，它是使用所有 Petit Fatfs 文件系统功能的前提）
mem_init();                                 //外部 SRAM 初始化
P1M0=0X00;    //P1 口为仅输入状态
P1M1=0Xff;
```

```
        /* 开启中断目的为判断 VSYNC（帧数据判断）：当有触发的时候为来一帧这时开始往摄像头 FIFO 缓存芯片 AL422B 灌入一帧数据。当来第二帧时说明数据灌入完成，此时提取数据进行显示。 */
        IT0=1;                  //边沿触发
        EX0=1;              //外部中断 0     P3.2 口
        EA =1;          //开总中断

        /* 将液晶显示屏用黑色清空 */
        GUI_Clear(Black);
        LCD_scan(1);  //液晶屏设置扫描方式：从上到下 从右到左
         /* OV7670 初始化 */
        if( Ov7670_init_normal()) //初始化为正常拍摄
        {                                                   //初始化不成功
         GUI_sprintf_hzstr16x(60,150,"OV7670 初始化",White,Blue);
         delay1ms(2000);
         EX0=0;                 //关闭外部中断 0
         EA=0;                  //关闭总中断
         sign=1;
        }

         if (SD_Init())                                     //SD 卡初始化，不成功退出
         {
             GUI_sprintf_hzstr16x(90,20,"TF Card Error!",Red, White);
             return;
         }
         else
         {
             GUI_sprintf_hzstr16x(90,20,"TF Card OK!",Black, White);
         }
        if(sign==1) return;       //初始化失败跳出 （没有插 OV7670 或者没插好）
        FIFO_OE=0;            //使能摄像头模块
        OV7670_Window_Set(10,176,240,240);     //设置显示窗口尺寸：240*240，场频是 10，行频是 176
        IO_init();                    //对 IAP15W4K61S4   IO 口初始化
         Lcd_Init();               //tft 初始化
         Init_SPI();                   //触摸屏 SPI 接口初始化
    while(1)
    {
        GUI_Clear(Black);
     /* 开始扫描 显示摄像头采集数据 */
     while(mode==0)//mode==0,参考线照相，五张连拍模式
     {
         scan();
        GUI_arc(120,269,25,White,1); //120，269 位置画一个半径为 25 的实心圆
        GUI_box(20,245,68,293,White);//左下角画一个 48*48 的实心白色矩形
        GUI_sprintf_hzstr16x(20,263,"replay", Red, White); //矩形的位置写黑色字
```

```
            GUI_line(80,0,80,240,White);//参考线
            GUI_line(160,0,160,240,White);
            GUI_line(0,80,240,80,White);
            GUI_line(0,160,240,160,White);
            GUI_tri(180,245,180,293,White);//三角形
            GetTouchScreenPos(&tp); //采集触摸屏
            if (tp.flag == 1)                   //是否有触摸事件发生
            {
                tp.flag = 0;
                if ((tp.x>20) && (tp.x<68) && (tp.y>245) && (tp.y<293)) //点击左下角矩形，进入回看
                {
                    led=0;
                    delay1ms(100);//延迟 0.1s
                    led=1;
                    mode=3;
                    delay1ms(100);
                    break;
                }

                    if ((tp.x>100) && (tp.x<140) && (tp.y>249) && (tp.y<288)) //点击圆形,再扫描一次（无参考线），并保存照片
                    {
                        for(j = 0;j<5;j++)//进行五次扫描，五连拍
                            {
                                scan();
                GUI_arc(120,269,25,White,1);        //120，269 位置画一个半径为 25 的实心圆
                GUI_box(20,245,68,293,White);//左下角画一个 48*48 的实心白色矩形
                GUI_sprintf_hzstr16x(20,263,"replay", Red, White); //矩形的位置写黑色字

                GUI_line(80,0,80,240,White);//参考线
                GUI_line(160,0,160,240,White);
                GUI_line(0,80,240,80,White);
                GUI_line(0,160,240,160,White);

                GUI_tri(180,245,180,293,White);//三角形
                                Save_Bmp(fname[current]);//每次拍照储存位置的指针+1
                                if(current+1>30)
                                {
                                    current=0;
                                }
                                else
                                {
                                    current++;
```

```
                        }
                    }
                }
            if ((tp.x>180) && (tp.x<220) && (tp.y>220) && (tp.y<295)) //点击三角形,改变模式
            {
                led=0;
                delay1ms(100);//延迟 0.1s
                led=1;
                if(mode+1<=3)
                {
                    mode ++;
    //这里进入了下一模式，重新初始化摄像头为负片滤镜
     EX0=1;              //外部中断 0     P3.2 口
      EA =1;          //开总中断

      /* 将液晶显示屏用黑色清空 */
      GUI_Clear(Black);
      LCD_scan(1);   //液晶屏设置扫描方式：从上到下 从右到左
                        if(Ov7670_init_fupian())
      {                                                          //初始化不成功
       GUI_sprintf_hzstr16x(60,150,"OV7670 初始化",White,Blue);
       delay1ms(2000);
       EX0=0;                 //关闭外部中断 0
       EA=0;                  //关闭总中断
       sign=1;
      }
       if (SD_Init())                              //SD 卡初始化，不成功退出
       {
           GUI_sprintf_hzstr16x(90,20,"TF Card Error!",Red, White);
           return;
       }
       else
       {
           GUI_sprintf_hzstr16x(90,20,"TF Card OK!",Black, White);
       }

      if(sign==1) return;          //初始化失败跳出 （没有插 OV7670 或者没插好）

      FIFO_OE=0;           //使能摄像头模块

      OV7670_Window_Set(10,176,240,240);     //设置显示窗口尺寸：240*240，场频是 10，行频是 176
                }
                else
                {
```

```
                    mode = 0;
                }
                    delay1ms(100);
                break;
            }

            }
        }

        while(mode==1)//mode==1,无参考线照相，负片滤镜模式
    {
        scan();
        GUI_arc(120,269,25,White,1);//120，269 位置画一个半径为 25 的实心圆
        GUI_box(20,245,68,293,White);//左下角画一个 48*48 的实心白色矩形
        GUI_sprintf_hzstr16x(20,263,"replay", Red, White); //矩形的位置写黑色字
        GUI_tri(180,245,180,293,White);//三角形
       //GUI_Target_PSO1(Black);//瞄准
        GetTouchScreenPos(&tp); //采集触摸屏
        if (tp.flag == 1)                   //是否有触摸事件发生
       {
            tp.flag = 0;
            if ((tp.x>20) && (tp.x<68) && (tp.y>245) && (tp.y<293)) //点击左下角矩形，进入回看
            {
                led=0;
                delay1ms(100);//延迟 0.1s
                led=1;
                mode=3;
                delay1ms(100);
                break;
            }
            if ((tp.x>100) && (tp.x<140) && (tp.y>249) && (tp.y<288)) //点击圆形,再扫描一次（无参考线），并保存
            {    Save_Bmp(fname[current]);
                 if(current+1>30)
                 {
                     current=0;
                 }
                 else
                 {
                     current++;
                 }
            }
            if ((tp.x>180) && (tp.x<220) && (tp.y>220) && (tp.y<295)) //点击三角形,改变模式
```

```
                {
                    led=0;
                    delay1ms(100);//延迟 0.1s
                    led=1;
                    if(mode+1<=3)
                    {
                        mode ++;
    //这里进入下一模式，重新初始化摄像头为反转
        EX0=1;              //外部中断 0     P3.2 口
        EA =1;          //开总中断

        /* 将液晶显示屏用黑色清空 */
        GUI_Clear(Black);
        LCD_scan(1);   //液晶屏设置扫描方式：从上到下 从右到左
                        if(Ov7670_init_fanzhuan())
        {                                                       //初始化不成
功
         GUI_sprintf_hzstr16x(60,150,"OV7670 初始化",White,Blue);
         delay1ms(2000);
         EX0=0;                 //关闭外部中断 0
         EA=0;                  //关闭总中断
         sign=1;
        }
         if (SD_Init())                             //SD 卡初始化，不成功退出
         {
             GUI_sprintf_hzstr16x(90,20,"TF Card Error!",Red, White);
             return;
         }
         else
         {
             GUI_sprintf_hzstr16x(90,20,"TF Card OK!",Black, White);
         }

        if(sign==1) return;          //初始化失败跳出 （没有插 OV7670 或者没插
好）

        FIFO_OE=0;          //使能摄像头模块

        OV7670_Window_Set(10,176,240,240);     //设置显示窗口尺寸：240*240，
场频是 10，行频是 176
                    }
                    else
                    {
                        mode = 0;
                    }
                        delay1ms(100);
                    break;
```

```
                }
            }
        }
            while(mode==2)//mode==2,瞄准线镜像反转照相
    {
            scan();
            GUI_arc(120,269,25,White,1);       //120，269 位置画一个半径为 25 的实心圆
            GUI_box(20,245,68,293,White);//左下角画一个 48*48 的实心白色矩形
            GUI_sprintf_hzstr16x(20,263,"replay", Red, White); //矩形的位置写黑色字
            GUI_tri(180,245,180,293,White);//三角形
          GUI_Target_PSO1(Black);//瞄准线水印绘制
            GetTouchScreenPos(&tp); //采集触摸屏
            if (tp.flag == 1)                  //是否有触摸事件发生
        {
                tp.flag = 0;
                if ((tp.x>20) && (tp.x<68) && (tp.y>245) && (tp.y<293)) //点击左下角矩形
                {
                    led=0;
                    delay1ms(100);//延迟 0.1s
                    led=1;
                    mode=3;
                    delay1ms(100);
                    break;
                }

                if ((tp.x>100) && (tp.x<140) && (tp.y>249) && (tp.y<288)) //点击圆形,再扫描一次（无参考线），并保存
                {
                    Save_Bmp(fname[current]);
                      if(current+1>30)
                        {
                            current=0;
                        }
                        else
                        {
                            current++;
                        }
                }
                if ((tp.x>180) && (tp.x<220) && (tp.y>220) && (tp.y<295)) //点击三角形,改变模式
                {
                    led=0;
                    delay1ms(100);//延迟 0.1s
```

```
                    led=1;
                    if(mode+1<=3)
                    {
                        mode ++;
                    }
                    else
                    {
                        mode = 0;
    //这里会进入模式 0，五连拍模式，需要重新初始化摄像头
      EX0=1;              //外部中断 0     P3.2 口
      EA =1;          //开总中断

      /* 将液晶显示屏用黑色清空 */
      GUI_Clear(Black);
      LCD_scan(1);   //液晶屏设置扫描方式：从上到下 从右到左
                    if(Ov7670_init_normal())
      {                                                          //初始化不成
功
       GUI_sprintf_hzstr16x(60,150,"OV7670 初始化",White,Blue);
       delay1ms(2000);
       EX0=0;                 //关闭外部中断 0
       EA=0;                  //关闭总中断
       sign=1;
      }
       if (SD_Init())                              //SD 卡初始化，不成功退出
       {
           GUI_sprintf_hzstr16x(90,20,"TF Card Error!",Red, White);
           return;
       }
       else
       {
           GUI_sprintf_hzstr16x(90,20,"TF Card OK!",Black, White);
       }

      if(sign==1) return;          //初始化失败跳出 （没有插 OV7670 或者没插
好）

      FIFO_OE=0;          //使能摄像头模块

      OV7670_Window_Set(10,176,240,240);     //设置显示窗口尺寸：240*240，
场频是 10，行频是 176
                    }
                        delay1ms(100);
                    break;
                }
            }
       }
```

```
    if(mode==3)//mode==3,回看界面
     {
          GUI_Clear(Black);
            GUI_box(20,245,68,293,White);//左下角画一个 48*48 的实心白色矩
形
            GUI_sprintf_hzstr16x(20,263,"home", Red, White); //矩形的位置写字
            GUI_box(96,245,144,293,White);//
            GUI_sprintf_hzstr16x(96,263,"last", Red, White); //矩形的位置写字
            GUI_box(172,245,220,293,White);       //120，269 位置画一个半径为
25 的实心圆
          GUI_sprintf_hzstr16x(172,263,"next", Red, White); //矩形的位置写字

           Show_Bmp(fname[current]); //显示第一张图片
         GUI_sprintf_hzstr16x(x, y, fname[current], Black, White);
                              x += 8*strlen(fname[current])+20;
                              number(x, y, pic_width, Black, White); //图片的宽度
                              x += 30;
                              GUI_sprintf_hzstr16x(x, y, "x", Black, White);
                              x += 16;
                              number(x, y, pic_height, Black, White); //图片的高度
        while(1){
            GetTouchScreenPos(&tp); //采集触摸屏
         if (tp.flag == 1)                  //是否有触摸事件发生
         {
                tp.flag = 0;
                if ((tp.x>20) && (tp.x<68) && (tp.y>245) && (tp.y<293)) //点击左
下角矩形，回到模式 0，重新初始化摄像头
                {
                    led=0;
                    delay1ms(100);//延迟 0.1s
                    led=1;
                    mode=0;
      EX0=1;              //外部中断 0     P3.2 口
      EA =1;          //开总中断

      /* 将液晶显示屏用黑色清空 */
      GUI_Clear(Black);
      LCD_scan(1);   //液晶屏设置扫描方式：从上到下 从右到左
                        if(Ov7670_init_normal())
      {                                                          //初始化不成
功
        GUI_sprintf_hzstr16x(60,150,"OV7670 初始化",White,Blue);
        delay1ms(2000);
        EX0=0;                  //关闭外部中断 0
        EA=0;                   //关闭总中断
        sign=1;
      }
```

```
        if (SD_Init())                          //SD 卡初始化，不成功退出
        {
            GUI_sprintf_hzstr16x(90,20,"TF Card Error!",Red, White);
            return;
        }
        else
        {
            GUI_sprintf_hzstr16x(90,20,"TF Card OK!",Black, White);
        }

      if(sign==1) return;          //初始化失败跳出 （没有插 OV7670 或者没插好）

      FIFO_OE=0;          //使能摄像头模块

      OV7670_Window_Set(10,176,240,240);     //设置显示窗口尺寸：240*240，场频是 10，行频是 176
                    delay1ms(100);
                    break;
                }
                if ((tp.x>96) && (tp.x<144) && (tp.y>245) && (tp.y<293)) //点击中间,切换照片
                {
                    GUI_Clear(Black);
            GUI_box(20,245,68,293,White);//左下角画一个 48*48 的实心白色矩形
            GUI_sprintf_hzstr16x(20,263,"home", Red, White); //矩形的位置写字
            GUI_box(96,245,144,293,White);//
            GUI_sprintf_hzstr16x(96,263,"last", Red, White); //矩形的位置写字
            GUI_box(172,245,220,293,White);       //120，269 位置画一个半径为 25 的实心圆
          GUI_sprintf_hzstr16x(172,263,"next", Red, White); //矩形的位置写字
                    led=0;
                    delay1ms(100);//延迟 0.1s
                    led=1;
                    if(current==1)
                        {
                            current=30;
                        }
                        else
                        {
                            current--;
                        }          //上一张
                     if(Show_Bmp(fname[current])==0)       //显示指定路径下的 bmp 文件
                                                //这是要已知存入 TF 卡里的路径"/ov76/M1.bmp"就是在 TF 卡根目录下文件名为机器猫.bmp 文件
```

```
                    {
                        x=0;
                        y=300;
                        GUI_sprintf_hzstr16x(x,  y,  fname[current],  Black,
White);
                        x += 8*strlen(fname[current])+20;
                        number(x, y, pic_width, Black, White); //图片的宽度
                        x += 30;
                        GUI_sprintf_hzstr16x(x, y, "x", Black, White);
                        x += 16;
                        number(x, y, pic_height, Black, White); //图片的高度
                    }

                }
                if ((tp.x>172) && (tp.x<220) && (tp.y>245) && (tp.y<293)) //点击
右下,切换照片
                {
                    GUI_Clear(Black);
            GUI_box(20,245,68,293,White);//左下角画一个 48*48 的实心白色矩
形
            GUI_sprintf_hzstr16x(20,263,"home", Red, White); //矩形的位置写字
            GUI_box(96,245,144,293,White);//
            GUI_sprintf_hzstr16x(96,263,"last", Red, White); //矩形的位置写字
            GUI_box(172,245,220,293,White);      //120，269 位置画一个半径为
25 的实心圆
          GUI_sprintf_hzstr16x(172,263,"next", Red, White); //矩形的位置写字
                    led=0;
                    delay1ms(100);//延迟 0.1s
                    led=1;
                     if(current==30)
                        {
                            current=0;
                        }
                        else
                        {
                            current++;
                        }               //下一张
                        if(Show_Bmp(fname[current])==0)        //显示指定路径
下的 bmp 文件
                                                    //这是要已知存入 TF 卡里的路径
"/ov76/M1.bmp"就是在 TF 卡根目录下文件名为机器猫.bmp 文件
                        {
                            x=0;
                            y=300;
                            GUI_sprintf_hzstr16x(x,  y,  fname[current],  Black,
White);
```

```
                        x += 8*strlen(fname[current])+20;
                        number(x, y, pic_width, Black, White); //图片的宽度
                        x += 30;
                        GUI_sprintf_hzstr16x(x, y, "x", Black, White);
                        x += 16;
                        number(x, y, pic_height, Black, White); //图片的高度
                    }
                }
            }
        }
    }
}
```

2. 自己写的 ov7670.h

```
#ifndef _OV7670_H_
#define _OV7670_H_
#include "def.h"



#define OV7670_REG_NUM    114


 //===========FIFO PIN===========
sbit FIFO_WEN =P3^3;            //写入 FIFO 使能
sbit FIFO_RCLK=P4^5;            //读数据时钟
sbit FIFO_WRST=P3^4;            //写指针复位
sbit FIFO_RRST=P3^5;            //读指针复位
sbit FIFO_OE    =P3^6;          //片选信号(OE)



extern u8 cur_status;                       //帧标志位    在 interrupt.c 函数中
调用



u8 wr_Sensor_Reg(u8 regID, u8 regDat);
u8 rd_Sensor_Reg(u8 regID, u8 *regDat);
```

u8 Ov7670_init_normal(void);　　//OV7670 初始化为五连拍模式

u8 Ov7670_init_fupian(void);//OV7670 初始化为负片模式

u8 Ov7670_init_fanzhuan(void); //OV7670 初始化为镜像模式

void OV7670_Window_Set(unsigned int sx,unsigned int sy,unsigned int width,unsigned int height);　　//ov7670 窗体设置

#endif

3.自己写的 OV7670.c

//摄像头模块 采用 OV7670 分辨率为 640x480 在实际应用中采用 QVGA 模式 即 240x320 模式

//对于黑灵接口 摄像头采集的图像和实际的图像是倒着的 但是采集框就是 240x320 大小 所以程序上采用 240x240 的显示框 显示正向的效果

//扫描方向为从左到右 从上到下

//这样做虽然少了一部分的显示 但是在显示速度上又加快了好多

```
#include  "STC/stc15f2k60s2.h"
#include  "gui.h"
#include  "tft.h"
#include  "ov7670.h"
#include  "delay.h"
#include  "sccb.h"
#include  "xpt2046.h"

//五连拍的 OV7670 寄存器配置
char code OV7670_reg_normal[OV7670_REG_NUM][2]=
{

    /*以下为 OV7670 QVGA RGB565 参数　　具体内容 可以参考资料里 ov7670 中文版 PDF 资料*/
    {0x3a, 0x04},//dummy
    {0x40, 0x10},//565
    {0x12, 0x14},//@
```

{0x17, 0x16},//行频开始高 8 位

{0x18, 0x04},//行频结束高 8 位

{0x19, 0x02},//场频开始高 8 位

{0x1a, 0x7a},//场频结束高 8 位

{0x32, 0x80},//行频开始低 3 位 bit[2:0]　行频结束低 3 位 bit[5:3]

{0x03, 0x0a},//位【3：2】场频结束低 2 位　位【1：0】场频开始低 2 位

{0x0c, 0x0c},

{0x15, 0x00},

{0x3e, 0x00},//10

{0x70, 0x00},

{0x71, 0x01},

{0x72, 0x11},

{0x73, 0x09},//

{0xa2, 0x02},//15

{0x11, 0x00}, //fenpin

{0x7a, 0x20},

{0x7b, 0x1c},

{0x7c, 0x28},

{0x7d, 0x3c},//20

{0x7e, 0x55},

{0x7f, 0x68},

{0x80, 0x76},

{0x81, 0x80},

{0x82, 0x88},

```
{0x83, 0x8f},
{0x84, 0x96},
{0x85, 0xa3},
{0x86, 0xaf},

{0x87, 0xc4},//30
{0x88, 0xd7},
{0x89, 0xe8},
{0x13, 0xe0},
{0x00, 0x00},//AGC

{0x10, 0x00},
{0x0d, 0x00},
{0x14, 0x20},//0x38, limit the max gain
{0xa5, 0x05},
{0xab, 0x07},

{0x24, 0x75},//40
{0x25, 0x63},
{0x26, 0xA5},
{0x9f, 0x78},
{0xa0, 0x68},

{0xa1, 0x03},//0x0b,
{0xa6, 0xdf},//0xd8,
{0xa7, 0xdf},//0xd8,
{0xa8, 0xf0},
{0xa9, 0x90},

{0xaa, 0x94},//50
{0x13, 0xe5},
{0x0e, 0x61},
{0x0f, 0x4b},
{0x16, 0x02},
```

```
    {0x1e, 0x37},//0x07,
{0x21, 0x02},
{0x22, 0x91},
{0x29, 0x07},
{0x33, 0x0b},

{0x35, 0x0b},//60
{0x37, 0x1d},
{0x38, 0x71},
{0x39, 0x2a},
{0x3c, 0x78},

{0x4d, 0x40},
{0x4e, 0x20},
{0x69, 0x5d},
{0x6b, 0x40},//PLL
{0x74, 0x19},
{0x8d, 0x4f},

{0x8e, 0x00},//70
{0x8f, 0x00},
{0x90, 0x00},
{0x91, 0x00},
{0x92, 0x00},//0x19,//0x66

{0x96, 0x00},
{0x9a, 0x80},
{0xb0, 0x84},
{0xb1, 0x0c},
{0xb2, 0x0e},

{0xb3, 0x82},//80
{0xb8, 0x0a},
```

```
  {0x43, 0x14},
  {0x44, 0xf0},
  {0x45, 0x34},

  {0x46, 0x58},
  {0x47, 0x28},
  {0x48, 0x3a},
  {0x59, 0x88},
  {0x5a, 0x88},

  {0x5b, 0x44},//90
  {0x5c, 0x67},
  {0x5d, 0x49},
  {0x5e, 0x0e},
  {0x64, 0x04},
  {0x65, 0x20},

  {0x66, 0x05},
  {0x94, 0x04},
  {0x95, 0x08},
  {0x6c, 0x0a},
  {0x6d, 0x55},


  {0x4f, 0x80},
  {0x50, 0x80},
  {0x51, 0x00},
  {0x52, 0x22},
  {0x53, 0x5e},
  {0x54, 0x80},

//{0x54, 0x40},//110
```

```
    //{0x09, 0x03},//驱动能力最大
      {0x6e, 0x11},//100
      {0x6f, 0x9f},//0x9e for advance AWB
       {0x55, 0x00},//亮度
       {0x56, 0x40},//对比度
       {0x57, 0x80},//0x40,    change according to Jim's request
};
//负片模式的寄存器设置
char code OV7670_reg_fupian[OV7670_REG_NUM][2]=
{

        /*以下为 OV7670 QVGA RGB565 参数       具体内容  可以参考资料里 ov7670 中文版 PDF 资料*/
      {0x3a, 0x24},//dummy
      {0x40, 0x10},//565
      {0x12, 0x14},//@


      {0x17, 0x16},//行频开始高 8 位
      {0x18, 0x04},//行频结束高 8 位
      {0x19, 0x02},//场频开始高 8 位
       {0x1a, 0x7a},//场频结束高 8 位

      {0x32, 0x80},//行频开始低 3 位 bit[2:0]     行频结束低 3 位 bit[5:3]
       {0x03, 0x0a},//位【3：2】场频结束低 2 位    位【1：0】场频开始低 2 位

      {0x0c, 0x0c},
       {0x15, 0x00},
      {0x3e, 0x00},//10
      {0x70, 0x00},
      {0x71, 0x01},
```

```
{0x72, 0x11},
{0x73, 0x09},//


{0xa2, 0x02},//15
   {0x11, 0x00}, //fenpin
{0x7a, 0x20},
{0x7b, 0x1c},
{0x7c, 0x28},


{0x7d, 0x3c},//20
{0x7e, 0x55},
{0x7f, 0x68},
{0x80, 0x76},
{0x81, 0x80},

{0x82, 0x88},
{0x83, 0x8f},
{0x84, 0x96},
{0x85, 0xa3},
{0x86, 0xaf},

{0x87, 0xc4},//30
{0x88, 0xd7},
{0x89, 0xe8},
{0x13, 0xe0},
{0x00, 0x00},//AGC

{0x10, 0x00},
{0x0d, 0x00},
{0x14, 0x20},//0x38, limit the max gain
{0xa5, 0x05},
{0xab, 0x07},
```

```
{0x24, 0x75},//40
{0x25, 0x63},
{0x26, 0xA5},
{0x9f, 0x78},
{0xa0, 0x68},

{0xa1, 0x03},//0x0b,
{0xa6, 0xdf},//0xd8,
{0xa7, 0xdf},//0xd8,
{0xa8, 0xf0},
{0xa9, 0x90},

{0xaa, 0x94},//50
{0x13, 0xe5},
{0x0e, 0x61},
{0x0f, 0x4b},
{0x16, 0x02},

   {0x1e, 0x37},//0x07,
{0x21, 0x02},
{0x22, 0x91},
{0x29, 0x07},
{0x33, 0x0b},

{0x35, 0x0b},//60
{0x37, 0x1d},
{0x38, 0x71},
{0x39, 0x2a},
{0x3c, 0x78},

{0x4d, 0x40},
{0x4e, 0x20},
{0x69, 0x5d},
```

```
{0x6b, 0x40},//PLL
{0x74, 0x19},
{0x8d, 0x4f},

{0x8e, 0x00},//70
{0x8f, 0x00},
{0x90, 0x00},
{0x91, 0x00},
{0x92, 0x00},//0x19,//0x66

{0x96, 0x00},
{0x9a, 0x80},
{0xb0, 0x84},
{0xb1, 0x0c},
{0xb2, 0x0e},

{0xb3, 0x82},//80
{0xb8, 0x0a},
{0x43, 0x14},
{0x44, 0xf0},
{0x45, 0x34},

{0x46, 0x58},
{0x47, 0x28},
{0x48, 0x3a},
{0x59, 0x88},
{0x5a, 0x88},

{0x5b, 0x44},//90
{0x5c, 0x67},
{0x5d, 0x49},
{0x5e, 0x0e},
{0x64, 0x04},
{0x65, 0x20},
```

```
        {0x66, 0x05},
        {0x94, 0x04},
        {0x95, 0x08},
        {0x6c, 0x0a},
        {0x6d, 0x55},


        {0x4f, 0x80},
        {0x50, 0x80},
        {0x51, 0x00},
        {0x52, 0x22},
        {0x53, 0x5e},
        {0x54, 0x80},

      //{0x54, 0x40},//110

      //{0x09, 0x03},//驱动能力最大
        {0x6e, 0x11},//100
        {0x6f, 0x9f},//0x9e for advance AWB
         {0x55, 0x00},//亮度
         {0x56, 0x40},//对比度
         {0x57, 0x80},//0x40,    change according to Jim's request
    };
    //镜像反转的寄存器设置
    char code OV7670_reg_fanzhuan[OV7670_REG_NUM][2]=
    {

          /*以下为 OV7670 QVGA RGB565 参数       具体内容  可以参考资料里
ov7670 中文版 PDF 资料*/
        {0x3a, 0x04},//dummy
        {0x40, 0x10},//565
        {0x12, 0x14},//@
```

{0x17, 0x16},//行频开始高 8 位

{0x18, 0x04},//行频结束高 8 位

{0x19, 0x02},//场频开始高 8 位

{0x1a, 0x7a},//场频结束高 8 位

{0x32, 0x80},//行频开始低 3 位 bit[2:0]　行频结束低 3 位 bit[5:3]

{0x03, 0x0a},//位【3：2】场频结束低 2 位　位【1：0】场频开始低 2 位

{0x0c, 0x0c},

{0x15, 0x00},

{0x3e, 0x00},//10

{0x70, 0x00},

{0x71, 0x01},

{0x72, 0x11},

{0x73, 0x09},//

{0xa2, 0x02},//15

{0x11, 0x00}, //fenpin

{0x7a, 0x20},

{0x7b, 0x1c},

{0x7c, 0x28},

{0x7d, 0x3c},//20

{0x7e, 0x55},

{0x7f, 0x68},

{0x80, 0x76},

{0x81, 0x80},

```
{0x82, 0x88},
{0x83, 0x8f},
{0x84, 0x96},
{0x85, 0xa3},
{0x86, 0xaf},

{0x87, 0xc4},//30
{0x88, 0xd7},
{0x89, 0xe8},
{0x13, 0xe0},
{0x00, 0x00},//AGC

{0x10, 0x00},
{0x0d, 0x00},
{0x14, 0x20},//0x38, limit the max gain
{0xa5, 0x05},
{0xab, 0x07},

{0x24, 0x75},//40
{0x25, 0x63},
{0x26, 0xA5},
{0x9f, 0x78},
{0xa0, 0x68},

{0xa1, 0x03},//0x0b,
{0xa6, 0xdf},//0xd8,
{0xa7, 0xdf},//0xd8,
{0xa8, 0xf0},
{0xa9, 0x90},

{0xaa, 0x94},//50
{0x13, 0xe5},
{0x0e, 0x61},
{0x0f, 0x4b},
```

```
{0x16, 0x02},

   {0x1e, 0x07},//0x07,
{0x21, 0x02},
{0x22, 0x91},
{0x29, 0x07},
{0x33, 0x0b},

{0x35, 0x0b},//60
{0x37, 0x1d},
{0x38, 0x71},
{0x39, 0x2a},
{0x3c, 0x78},

{0x4d, 0x40},
{0x4e, 0x20},
{0x69, 0x5d},
{0x6b, 0x40},//PLL
{0x74, 0x19},
{0x8d, 0x4f},

{0x8e, 0x00},//70
{0x8f, 0x00},
{0x90, 0x00},
{0x91, 0x00},
{0x92, 0x00},//0x19,//0x66

{0x96, 0x00},
{0x9a, 0x80},
{0xb0, 0x84},
{0xb1, 0x0c},
{0xb2, 0x0e},

{0xb3, 0x82},//80
```

```
{0xb8, 0x0a},
{0x43, 0x14},
{0x44, 0xf0},
{0x45, 0x34},

{0x46, 0x58},
{0x47, 0x28},
{0x48, 0x3a},
{0x59, 0x88},
{0x5a, 0x88},

{0x5b, 0x44},//90
{0x5c, 0x67},
{0x5d, 0x49},
{0x5e, 0x0e},
{0x64, 0x04},
{0x65, 0x20},

{0x66, 0x05},
{0x94, 0x04},
{0x95, 0x08},
{0x6c, 0x0a},
{0x6d, 0x55},


{0x4f, 0x80},
{0x50, 0x80},
{0x51, 0x00},
{0x52, 0x22},
{0x53, 0x5e},
{0x54, 0x80},

//{0x54, 0x40},//110
```

```
  //{0x09, 0x03},//驱动能力最大
    {0x6e, 0x11},//100
    {0x6f, 0x9f},//0x9e for advance AWB
     {0x55, 0x00},//亮度
     {0x56, 0x40},//对比度
     {0x57, 0x80},//0x40,  change according to Jim's request
};




//写 OV7660 寄存器
//写 OV7660 寄存器
//返回：1-成功 0-失败
//regID   寄存器地址   regDat   数据
u8 wr_Sensor_Reg(u8 regID, u8 regDat)
{

 startSCCB(); //发送 SCCB  总线开始传输命令

 if(SCCBwriteByte(0x42)==0)//写地址          手册 11 页上
 {
  stopSCCB();//发送 SCCB  总线停止传输命令
  return(0);//错误返回
 }

 if(SCCBwriteByte(regID)==0)//寄存器 ID
 {
  stopSCCB();//发送 SCCB  总线停止传输命令
  return(0); //错误返回
 }
```

```
  if(SCCBwriteByte(regDat)==0)//写数据到积存器
  {
   stopSCCB();//发送 SCCB  总线停止传输命令
   return(0);//错误返回
  }

  stopSCCB();//发送 SCCB  总线停止传输命令


  return(1);//成功返回
}


//读 OV7660 寄存器
//返回：1-成功 0-失败
//regID   寄存器地址    regDat    数据
u8 rd_Sensor_Reg(u8 regID, u8 *regDat)
{
  //通过写操作设置寄存器地址

  startSCCB();

  if(0==SCCBwriteByte(0x42))//写地址          手册 11 页上
  {
   stopSCCB();//发送 SCCB  总线停止传输命令
   return(0);//错误返回
  }

  if(0==SCCBwriteByte(regID))//积存器 ID
  {
   stopSCCB();//发送 SCCB  总线停止传输命令
   return(0);//错误返回
```

```
  }
  stopSCCB();//发送 SCCB  总线停止传输命令

  //设置寄存器地址后，才是读
  startSCCB();

  if(0==SCCBwriteByte(0x43))//读地址              手册 11 页上
  {
   stopSCCB();//发送 SCCB  总线停止传输命令
   return(0);//错误返回
  }

  *regDat=SCCBreadByte();//返回读到的值

  noAck();//发送 NACK 命令
  stopSCCB();//发送 SCCB  总线停止传输命令

  return(1);//成功返回
}

//五连拍模式初始化
//Sensor_init()   摄像头芯片初始化
//返回 0 成功，返回 1 失败
//=================================================
u8 Ov7670_init_normal(void)
{
  u8    i=0,
        tem1,tem2;


  if(0==wr_Sensor_Reg(0x12,0x80)) //Reset SCCB 复位 SCCB
  {
```

```
  return 1 ;//错误返回
 }

 if(0==rd_Sensor_Reg(0x0b, &tem1))//读 ID  产品低字节识别号
 {
  return 1 ;//错误返回
 }
  if(0==rd_Sensor_Reg(0x0a, &tem2))//读 ID 产品高字节识别号
 {
  return 1 ;//错误返回
 }


 if(tem1==0x73)//OV7670
 {
  if(tem2==0x76)
  {
  for(i=0;i<OV7670_REG_NUM;i++)                    //写寄存器循环
  {


if(0==wr_Sensor_Reg(OV7670_reg_normal[i][0],OV7670_reg_normal[i][1]))
     {
      return 1;//错误返回
     }
  }
  return 0;
 }
 }
return 1;   //错误返回
}
```

```
//镜像模式初始化
u8 Ov7670_init_fanzhuan(void)
{
  u8    i=0,
        tem1,tem2;


  if(0==wr_Sensor_Reg(0x12,0x80)) //Reset SCCB 复位 SCCB
  {
   return 1 ;//错误返回
  }

  if(0==rd_Sensor_Reg(0x0b, &tem1))//读 ID  产品低字节识别号
  {
   return 1 ;//错误返回
  }
   if(0==rd_Sensor_Reg(0x0a, &tem2))//读 ID 产品高字节识别号
  {
   return 1 ;//错误返回
  }


  if(tem1==0x73)//OV7670
  {
   if(tem2==0x76)
   {
   for(i=0;i<OV7670_REG_NUM;i++)                  //写寄存器循环
   {


if(0==wr_Sensor_Reg(OV7670_reg_fanzhuan[i][0],OV7670_reg_fanzhuan[i][1]))
     {
```

```
        return 1;//错误返回
        }
      }
      return 0;
    }
    }
  return 1;   //错误返回
  }

  //负片模式初始化
//Sensor_init()   摄像头芯片初始化
  //返回 0 成功，返回 1 失败
  //==================================================
  u8 Ov7670_init_fupian(void)
  {
   u8    i=0,
         tem1,tem2;


   if(0==wr_Sensor_Reg(0x12,0x80)) //Reset SCCB 复位 SCCB
   {
    return 1 ;//错误返回
   }

   if(0==rd_Sensor_Reg(0x0b, &tem1))//读 ID  产品低字节识别号
   {
    return 1 ;//错误返回
   }
    if(0==rd_Sensor_Reg(0x0a, &tem2))//读 ID 产品高字节识别号
   {
    return 1 ;//错误返回
   }
```

```
 if(tem1==0x73)//OV7670
 {
  if(tem2==0x76)
  {
  for(i=0;i<OV7670_REG_NUM;i++)                    //写寄存器循环
  {

    if(0==wr_Sensor_Reg(OV7670_reg_fupian[i][0],OV7670_reg_fupian[i][1]))
    {
    return 1;//错误返回
    }
  }
  return 0;
 }
 }
return 1;   //错误返回
}
//设置图像输出窗口
//对 QVGA 设置。
//sx   sy  为场频  和行频的初始值  这个值是 ov 本身厂家定的  在初始化里计数一下就能知道  场频是 10   行频是 176
//width  为场频方向     height  为行频方向
//摄像头在初始的时候  就已经设置成 QVGA 格式  也就是说是 320x240 的分辨率
//那么在设置显示区域的时候不能超过这个个分辨率  否则也不会有多余的显示  注意!
void OV7670_Window_Set(u16 sx,u16 sy,u16 width,u16 height)
{
    u16 endx;
    u16 endy;
```

```
    u8 temp;
    endx=sx+width*2;    //V*2     一个颜色两个字节
    endy=sy+height*2;
    if(endy>784)endy-=784;        //这里要看时序图  784 为行频的一个周期


    rd_Sensor_Reg(0X03,&temp);      //读取场频低字节
    temp&=0XF0;                     //清低 4 位
    temp|=((endx&0X03)<<2)|(sx&0X03);   //将计算的低字节移入

    wr_Sensor_Reg(0X03,temp);         //写入
    wr_Sensor_Reg(0X19,sx>>2);
    wr_Sensor_Reg(0X1A,endx>>2);

    rd_Sensor_Reg(0X32,&temp);              //同上
    temp&=0XC0;
    temp|=((endy&0X07)<<3)|(sy&0X07);

    wr_Sensor_Reg(0X32,temp);
     wr_Sensor_Reg(0X17,sy>>3);           //行频开始高 8 位
    wr_Sensor_Reg(0X18,endy>>3);    //行频结束高 8 位
}

u8 cur_status=0;              //帧标志位    在 interrupt.c 函数中调用

void zhen() interrupt 0 //外部中断 0    P3.2    判断帧数据
 {


    if(cur_status==0) //如果此时状态为 0，则说明是一个图像的开始，开始向
FIFO 罐入数据
```

```
        {
         FIFO_WRST=0;     //写复位
         FIFO_WRST=1;
         FIFO_WEN=1;      //写 FIFO 使能
         cur_status=1;    //标记为 1
        }
       else
        if(cur_status==1)   //说明此处为图像的结束，亦即下一图像的开始
         {
           FIFO_WEN=0;       //写 FIFO 禁能
           FIFO_WRST=0;          //写复位
           FIFO_WRST=1;
           cur_status=2;   //标记为 2   此时说明可以读取 FIFO 中的缓存数据
         }
      }
```

3. 自己写的 GUI.C

```
#include   "stc15f2k60s2.h"
#include   "gui.h"
#include   "tft.h"
#include   "delay.h"
#include   "zifu8x16.h"
#include   "hz16x16.h"          //汉字 16X16
#include   "pff.h"
#include   "sd.h"
#include   "flash.h"
```

//在指定位置显示 字库里的 GBK 汉字　　当字体背景颜色 只是 0x0001 那么此时不显示背景颜色 背景颜色为默认颜色

//支持横向纵向显示选择功能 mode　但是只有纵向显示的时候才支持 背景颜色保持屏幕本身有的颜色

// x　y　显示的具体坐标

// Disp_char[2] 需要显示的 GBK 码

```
// fColor       bColor       字体 背景颜色
// mode       0 纵向显示   1 横向显示   （主要要配合扫描方式）
//返回 0 显示成功   返回 1 在字库内 没有对应的 GBK 码
u8 PutGB1616(u16 x, u16   y, u8 Disp_char[2], u16 fColor,u16 bColor,u8 mode)
{

    u8 qh,ql;
    u8 i,j;
    u8   tmp_Char_Model[32];   //GBK12 点阵字库中   提取一个 16X16 点阵 即 32 个字节

    qh=Disp_char[0];
    ql=Disp_char[1];

    if(qh<0x81||ql<0x40||ql==0xff||qh==0xff)//非 常用汉字
    return 1; //结束访问

    if(ql<0x7f)ql-=0x40;//根据低字节在两个区域 将高低字节转移到 指定字节位置
    else ql-=0x41;
    qh-=0x81;


    // 读字模                          //   得到区值          得到位数值
   一个字 32 个字节
    //SD_read_Byte((gbk12_sector<<9)      +      ((u32)qh*190      +
(u32)ql)*32,tmp_Char_Model,32);     //TF 卡 读取模式

    /*
     TF 卡模式
     gbk12_sector 为字库所在扇区     gbk12_sector<<9 相当于 X512   转为字节数据 即字库所在字节开始位置
```

每个 GBK 码由 2 个字节组成，

FLASH 模式

flash 中已经将 TF 卡里的 字库数据完整存入 flash 中 只要确定地址即可 这个地址就是存入时的第一个地址

这里是从 第 322 个扇区开始存入的 flash 的一个扇区是 4k 的字节空间 即 4096 个字节 所以首地址是 322*4096

第一个字节为 0X81~0XFE，

第二个字节分为两部分， 一是 0X40~0X7E，二是 0X80~0XFE 。

第一个字节代表为区，那么 GBK 里面总共有 126 个区（0XFE-0X81+1 ），

每个区内有 190 个汉字（0XFE-0X80+0X7E-0X40+2 ），总共就有 126*190=23940 个汉字

点阵库编码规则从 0X8140 开始 所以首先要判断低字节所在区域 然后得到具体字节位置

((u32)qh*190 + (u32)ql)*32 每个区 190 个字 一个字 32 个字节

*/

SPI_Flash_Read(tmp_Char_Model,(u32)322*4096+((u32)qh*190 + (u32)ql)*32,32); //FLASH 读取模式

if(bColor!=0x0001) //支持背景颜色设置

{ //注意 不管设置哪一显示方向 前提是扫描要对应

if(mode==0)Address_set(x,y, x+16-1, y+16-1); // 设置为纵向显示

if(mode==1)Address_set(x,y, x+16-1, y+16-1);// 设置为纵向显示

//显示出来

for(i=0;i<32;i++)

```
            {

                for(j=0;j<8;j++)
                {
                    if(tmp_Char_Model[i] & 0x80)
                            {
                               Lcd_Write_Data(fColor);
                            }
                    else
                             {
                               Lcd_Write_Data(bColor);
                            }
                    tmp_Char_Model[i] <<= 1;
                }
             }
          }
        else                                        //不支持背景颜色设置   只显示
字体颜色  背景颜色与刷屏时一致
         {
            qh=x;                                   //复制 x 值给 qh
           for(i=0;i<32;i++)
           {
                for(j=0;j<8;j++)
                {
                  Address_set(x,y,x,y);//显示坐标要一个点一个点显示
                    if(tmp_Char_Model[i] & 0x80)
                            {
                               Lcd_Write_Data(fColor);      //显示汉字的颜色
                              x++;                          //x 自加
                              if((x-qh)==16){x=qh;y++;}     //到一行底  清 x 值
                            }
                    else
                             {                              //没有汉字的颜色
```

要显示 即背景 背景不显示使用屏幕自带

```
                                    x++;
                                    if((x-qh)==16){x=qh;y++;}    //到一行底 清 x 值
                                }
                            tmp_Char_Model[i] <<= 1;                //下一个字节
                        }
                    }

            }

    return 0;//成功

}
//清屏
//color 是背景颜色。
//说明：使用背景颜色清除 TFT 模块屏幕的全部显示内容。
void GUI_Clear(u16 color)
{
    u16 i;
    u8 j;
    Address_set(0,0,239,319);
     for(i=0;i<320;i++)
      {
        for (j=0;j<240;j++)
          {
              Lcd_Write_Data(color);
           }

      }
}
//画点
//(x，y)是点的坐标
```

```
//color 是点的颜色。
//说明：用指定的颜色在指定的坐标位置上画出一个点。
void GUI_Point(u8 x, u16 y, u16 color)
{
    Address_set(x,y,x,y);
    Lcd_Write_Data(color);
}

void GUI_line(u16 x1,u16 y1,u16 x2,u16 y2,u16 color)//画直线，要求 x1,yi 为起点坐标，x2,y2 终点坐标。必须有一个坐标相等且终点坐标值要更大
{
    int i;
    if(x1 == x2)
    {
        for(i = y1;i < y2; i++)
        {
            GUI_Point(x1, i, color);
        }
    }
    if(y1 == y2)
    {
        for(i = x1;i < x2; i++)
        {
            GUI_Point(i, y1, color);
        }
    }
}

//以某条垂直线为底，在其右侧画等腰三角形
void GUI_tri(u16 x1,u16 y1,u16 x2,u16 y2,u16 color)//画三角形
{
    u16 len;
    int i;
```

```
    int tempy = y2;
    int tempx = x1;
    len = y2 - y1;//垂直线段长度
    for(i = len;i >=1; i=i-2)
    {

        GUI_line(tempx,tempy-i,tempx,tempy,color);
        tempy = tempy-1;
        tempx = tempx+1;
    }
}


//在指定位置画一个指定大小的圆
//(rx,ry):圆心
//r     :半径
//color:颜色
//mode :0,不填充;1,填充
void GUI_arc(u16 rx,u16 ry,u16 r,u16 color,u8 mode)
{
    int a,b,c;
    int di;
    a=0;b=r;
    di=3-(r<<1); //判断下个点位置的标志
    while(a<=b)
    {
        if(mode)
        for(c=a;c<=b;c++)
        gui_circle8(rx,ry,a,c,color);//画实心圆
        else gui_circle8(rx,ry,a,b,color);          //画空心圆
        a++;
        //使用 Bresenham 算法画圆
```

```
            if(di<0)di +=4*a+6;
            else
            {
                di+=10+4*(a-b);
                b--;
            }
        }
}

//画实心矩形
//(sx,sy)左上角顶点坐标,
//(ex,ey)右下角顶点坐标, color 颜色
//返回: 无
//说明：在指定位置上画出实心矩形。
void GUI_box(u8 sx,u16 sy,u8 ex,u16 ey,u16 color)
{
    u16 temp,temp1,m,n;
    Address_set(sx,sy,ex,ey);
     n=ex-sx+1;
    m=ey-sy+1;

    for(temp=0;temp<m;temp++)
    {
        for(temp1=0;temp1<n;temp1++)
        {
            Lcd_Write_Data(color);
        }
    }
}


//填充矩形
//x0,y0:矩形的左上角坐标
```

```
//width,height:矩形的尺寸
//color:颜色
void GUI_fill_box(u16 x0,u16 y0,u16 width,u16 height,u16 color)
{
    if(width==0||height==0)return;//非法.
    GUI_box(x0,y0,x0+width-1,y0+height-1,color);
}

//显示汉字字符串  纵向显示
//x1 y1  显示的初始位置
//*str  要显示的数据
//dcolor  显示字符的颜色
//bgcolor  显示字符的背景颜色
void GUI_sprintf_hzstr16x(u16 x1,u16 y1,u8 *str,u16 dcolor,u16 bgcolor)
{
     u8 l=0;
    while(*str)
    {
      if(*str<0x80)            //小于 128     ascii  都在数组内
      {
        GUI_sprintf_char(x1+l*8,y1,*str,dcolor,bgcolor,0);
        l+=1;
        str++;
        }
      else
      {
       PutGB1616(x1+l*8,y1,(u8*)str,dcolor, bgcolor,0);
            str+=2;l+=2;
      }
    }
}
    int i,j,k;
```

```
for (j=100;j<102;j++)    //水平准线
{
    for (i = 25; i < 80; i++)       //水平准线  左
    {
        GUI_Point(i,j,color);
    }
    for ( i = 160; i < 215; i++)        //水平准线  右
    {
        GUI_Point(i,j,color);
    }
}

for (i=90;i<=150;i=i+10)     //横向竖直分划线
{
    if (i!=120)
    {
        for (j=100;j<105;j++)
        {
            GUI_Point(i,j,color);
        }
    }
}

for (i=117;i<123;i++)     //三角准心
{
    if (i<120)
    {
        j=219-i;
        for (k=0;k<3;k++)
        {
            GUI_Point(i,j,color);
            j=j+10;
```

```
                }
            }
            else
            {
                j=i-20;
                for (k=0;k<3;k++)
                {
                    GUI_Point(i,j,color);
                    j=j+10;
                }
            }
        }

        for (i=119;i<121;i++)   //竖直准线  下
        {
            for (j=140;j<200;j++)
            {
                GUI_Point(i,j,color);
            }
        }
    }
```

4. 自己写的 GUI.h

```
#ifndef GUI_H
#define GUI_H
#include "def.h"

/*GUI 函数*/
void GUI_Clear(u16 color);    //清屏

void GUI_Point(u8 x, u16 y, u16 color);//画点

void GUI_line(u16 x1,u16 y1,u16 x2,u16 y2,u16 color)    ;//画直线
```

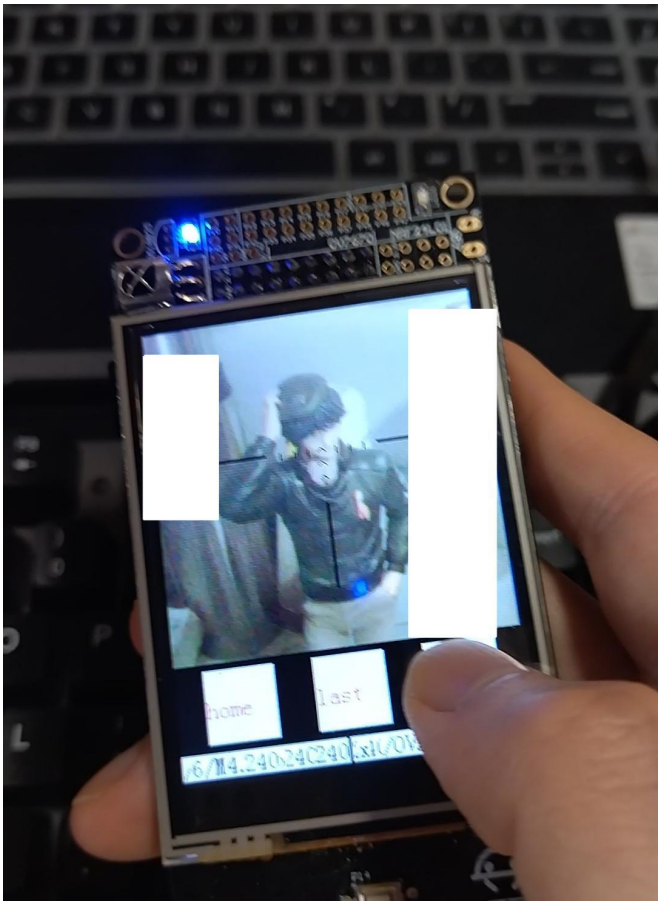
```
void GUI_tri(u16 x1,u16 y1,u16 x2,u16 y2,u16 color)    ;//画三角形
void GUI_sprintf_char(u16 x,u16 y,u8 value,u16 dcolor,u16 bgcolor,u8 mode); //显示英文或数字字符

void GUI_box(u8 sx,u16 sy,u8 ex,u16 ey,u16 color);//画实心矩形
void    GUI_sprintf_hzstr16x(u16 x1,u16 y1,u8 *str,u16 dcolor,u16 bgcolor);//显示汉字及字符    纵向显示

void GUI_arc(u16 rx,u16 ry,u16 r,u16 color,u8 mode);                    //指定位置画圆
void GUI_Target_PSO1(u16 color);        // PSO-1 形式瞄准线

u8 get_font_sector(void);                                               //提取 TF 卡中    GBK 点阵码    首扇区
/*定义常用颜色码*/
#define Red           0xf800      //红
#define Yellow      0xffe0 //黄
#define Green       0x07e0      //绿
#define Cyan         0x07ff      //青
#define Blue         0x001f//蓝
#define Purple      0xf81f  //紫
#define Black       0x0000      //黑
#define White       0xffff  //白
#define Gray         0x7bef      //灰
#define Blue1       0xa5ff //淡蓝
#define Blue2       0x7cdf
#define Purple1     0x8a9e //淡紫
#define Green1      0x0410 //墨绿
#define Green2      0x2616
#define Blue3       0x751E
#define Purple2     0xcd9e //淡紫
```

```
#define Red2        0XF260      //淡红
#define Red3        0X8000      //棕红
#define window      0XDED7
#endif
```

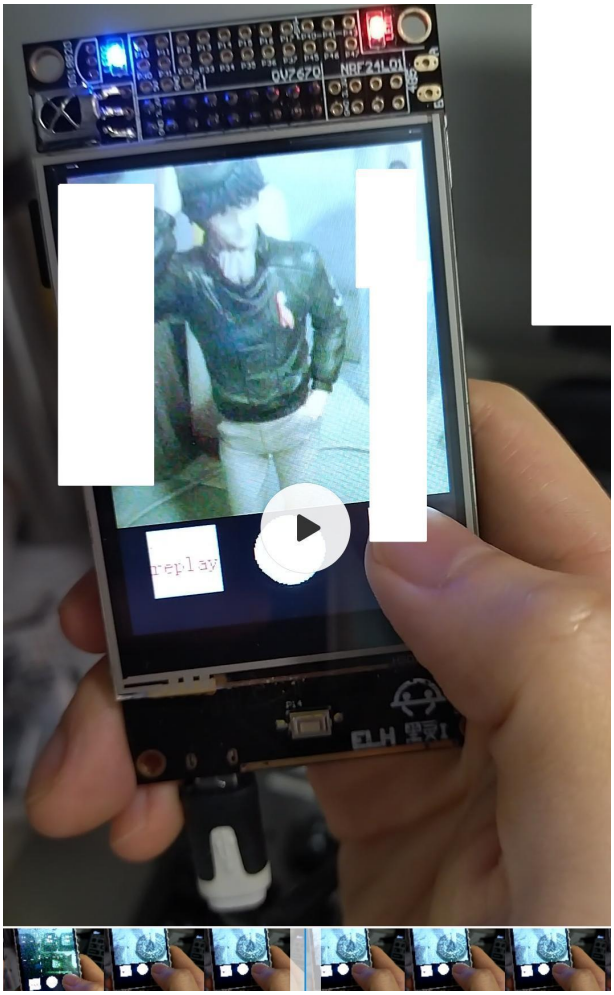
# 五、结果与分析

（要给出实际拍摄的最好**包含有本人头像的照片效果**和**实物图**）

## 0. 界面设置




**回看界面**左下矩形上写有“home”，回到照相模式
中间矩形上写有“last”，可看上一张照片
右下矩形上写有“next”，可看下一张照片
最下方的文本显示的是图片的名称的大小




**拍照界面**

左下矩形上写有“replay”，可进入回看界面

中间的圆形可以拍照

右下的三角形可以切换到下一模式，如果是最后一个模式（负片），则再按下后会回到第一个模式（五连拍）

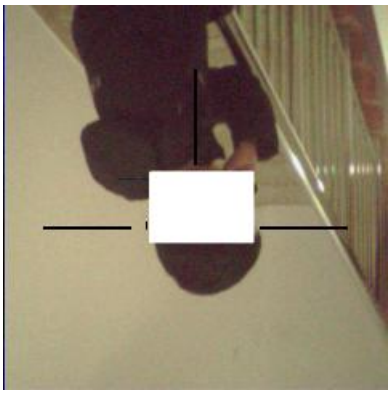
## 1.头像照片

头像，带参考线

## 2.五连拍模式下物品

物品，带参考线

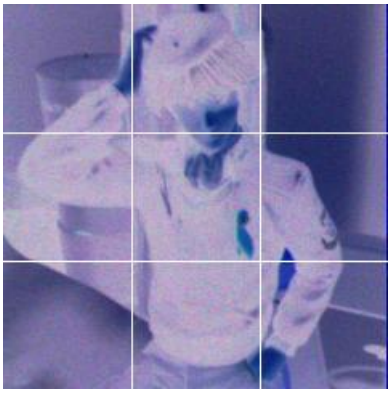
**3.镜像模式下人像与物品**



人像，带瞄准线水印

品，带瞄准线

**4.负片模式下物品**

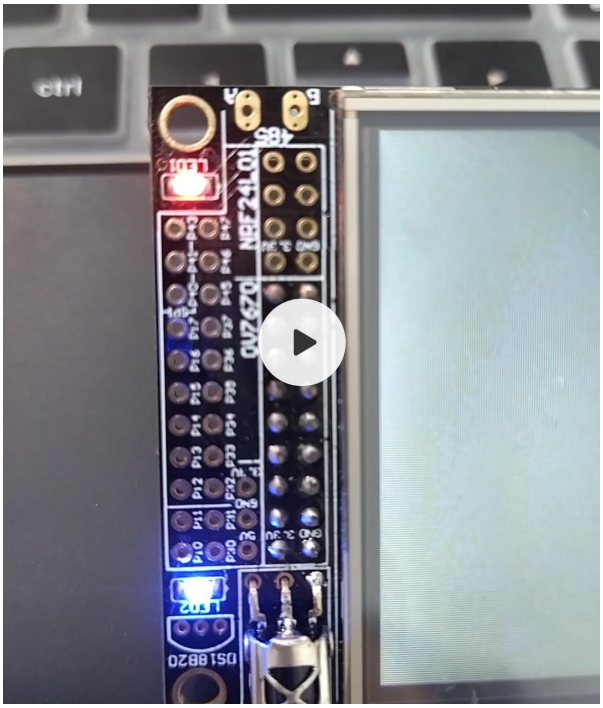


物品，带参考线，反色滤镜（摄像头 3a 地址的寄存器第 5 位置 1）

物品，带瞄准线水印，反色滤镜（摄像头 3a 地址的寄存器第 5 位置 1）

# 六、五个实验结果

## 1.P3.5 口 led 灯控制




效果为 led 每 0.1s 闪烁一次