**鲁东大学信息与电气工程学院2019－2020学年第 1 学期**

**学院\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_专业\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_班级\_\_\_\_\_\_\_\_\_\_\_\_\_本专 学号\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_姓名\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**密封线 学生须将文字写在此线以下**

**《嵌入式系统与应用》课程设计**

**课程号：220062040**

**任课教师** 贾世祥 **成绩**

|  |
| --- |
| **论文题目：**  智能电动百叶窗的设计与实现 |
| **论文要求：**（对论文题目、内容、行文、字数等作出判分规定。）  论文中首先对系统的功能的设计思路做出介绍，然后给出系统的工作原理、电路原理图、实物连接图，最后对软件的关键代码给出详细说明。要求图表清楚、大小合适，字数2000字左右。 |
| **教师评语：**  **教师签字：**  **年 月 日** |

智能电动百叶窗的设计与实现

李昂，20172202798

向世城，20172202833

1．系统设计思路

本课程设计实现了一个智能窗帘。用户使用三个按键控制窗帘的模式切换。一个按键操控智能窗帘的手动与自动模式切换，通过一个发光二极管显示模式，自动状态下二极管亮，在手动模式下，通过另外两个按键实现窗帘的关、停；在自动模式下，通过光敏电阻感测室内光强度，光照过强窗帘关闭，正常停止关闭；并通过串口将每个模式下的状态打印到电脑的超级终端上。

2．系统硬件实现

程序刚开始运行，对系统的时钟进行初始化，使能GPIO、AFIO引脚和USART1串口；设置外部中断15-10和9-5为中断触发方式；外部中断线路通过GPIO管脚触发，设置PC13、PG15、PF7引脚作为外部中断线13、外部中断线15、外部中断线7触发外部中断，使能沿下边沿触发；在设置GPIO管脚设置，USART发送管脚PA9复用推挽、接受引脚PA10为浮空输入，5个LED灯为推挽输出，三个按键为浮空输入，电机PB14推挽输出，光敏电阻PC2为浮空输入；串口通信波特率115200，８位数据位，停止位为１，无奇偶校验，硬件流控制失能，发送、接受功能并行。

初始化完成之后进入程序，先熄灭所有LED灯，进入while循环，按下key0键开始进入自动状态，LED1点亮，此后每次按下key0状态都会进行切换，并在切换成手动状态时熄灭LED1。在自动状态下，芯片的PC2管脚会时刻接收光敏电阻返回值，当光照过强时，PC2管脚电平为1，此时芯片将PB14引脚的电平赋为1，电机开始转动，串口发送“Auto light”，光照较暗时，PC2管脚电平为0，此时芯片将PB14引脚的电平赋为0，电机停止转动，串口发送“Auto dick”。在自动状态时，先将LED1赋0，熄灭LED，串口发送“Manual”，等待下一步命令，此时按下key1键，芯片将PB14引脚的电平赋为1，电机开始在手动状态下转动，串口发送“Manual open”，按下key2键，芯片将PB14引脚的电平赋为0，电机开始在手动状态下停止转动，串口发送“Manual close”。



图1 实验箱实物图

上图为实验箱图片，内部所有线路都已配置好，不需要手动连接，只需在程序中更改设置，开启专用引脚的复用功能即可。

3．系统软件实现

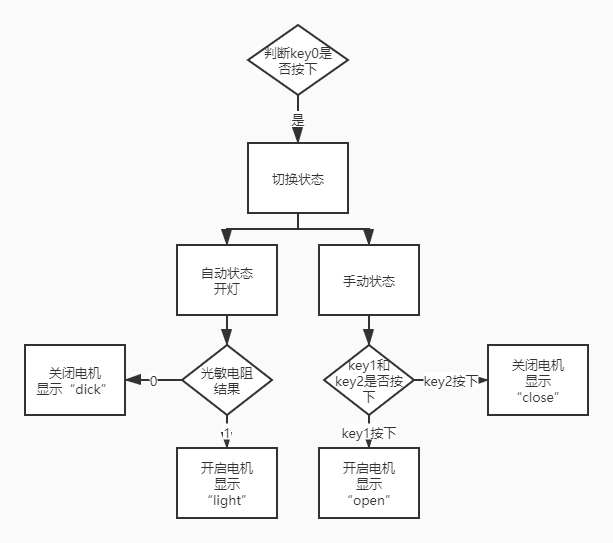


图2 软件流程图

系统软件部分主要分为六个大模块，主函数，RCC时钟设置，中断向量控制器设置，外部中断设置，GPIO口设置，串口设置。

在程序开始时，我们先声明头文件

#include "stm32f10x\_lib.h"

#include "platform\_config.h"

在"platform\_config.h"头文件中我们进行各项参数的设置

#define keypad\_0 0x01

#define keypad\_1 0x02

#define keypad\_2 0x03

#define keypad\_3 0x04

#define GPIO\_LED1 GPIOC

#define GPIO\_PIN\_LED1 GPIO\_Pin\_4

#define GPIO\_LED2 GPIOC

#define GPIO\_PIN\_LED2 GPIO\_Pin\_5

#define GPIO\_LED3 GPIOB

#define GPIO\_PIN\_LED3 GPIO\_Pin\_0

#define GPIO\_LED4 GPIOB

#define GPIO\_PIN\_LED4 GPIO\_Pin\_1

#define GPIO\_LED5 GPIOF

#define GPIO\_PIN\_LED5 GPIO\_Pin\_11

以及定义各种功能函数

void RCC\_Configuration(void);

void NVIC\_Configuration(void);

void GPIO\_Configuration(void);

void USART\_Configuration(void);

void Delaytime(u32 nCount);

void EXTI\_Configuration(void);

void Light\_up\_LED(GPIO\_TypeDef\* GPIOx , u16 GPIO\_Pin);

在"stm32f10x\_lib.h"头文件中，我们编写了3个外部中断程序

（1）该中断在手动状态（flag=0）下判断key2是否按下

void EXTI9\_5\_IRQHandler(void)

{

if(EXTI\_GetITStatus(EXTI\_Line7) != RESET)

{

Delaytime(0xF000);

if(!GPIO\_ReadInputDataBit(GPIOF,GPIO\_Pin\_7)&&flag==0)

{

KEYPAD\_STATE=keypad\_2;

}

/\* Clear the Key Button EXTI line pending bit \*/

EXTI\_ClearITPendingBit(EXTI\_Line7);

}

}

（2）该中断在手动状态（flag=0）下判断key1是否按下

void EXTI15\_10\_IRQHandler(void)

{

if(EXTI\_GetITStatus(EXTI\_Line15) != RESET)

{

Delaytime(0xF000);

if(!GPIO\_ReadInputDataBit(GPIOG,GPIO\_Pin\_15)&&flag==0)

{

KEYPAD\_STATE=keypad\_1;

}

/\* Clear the Key Button EXTI line pending bit \*/

EXTI\_ClearITPendingBit(EXTI\_Line15);

}

（3）该中断控制key0转换（手动/自动），并将key0设置了更高的优先级。

if(EXTI\_GetITStatus(EXTI\_Line13) != RESET)

{

Delaytime(0xF000);

if(!GPIO\_ReadInputDataBit(GPIOC,GPIO\_Pin\_13))

{

flag=!flag;

if(flag==1)

{

KEYPAD\_STATE=keypad\_0;

}

else

{

KEYPAD\_STATE=0x04;

}

}

/\* Clear the Key Button EXTI line pending bit \*/

EXTI\_ClearITPendingBit(EXTI\_Line13);

}

}

下面进入到主函数

int main(void)

{

#ifdef DEBUG

debug();

#endif

//各种功能函数初始化

/\* System Clocks Configuration \*/

RCC\_Configuration();

/\* NVIC Configuration \*/

NVIC\_Configuration();

/\* Configure the GPIO ports \*/

GPIO\_Configuration();

/\* Configure the USART \*/

USART\_Configuration();

/\* EXTI Configure \*/

EXTI\_Configuration();

/\*先熄灭所有的LED灯\*/

GPIO\_ResetBits(GPIOC, GPIO\_Pin\_4);

GPIO\_ResetBits(GPIOC, GPIO\_Pin\_5);

GPIO\_ResetBits(GPIOB, GPIO\_Pin\_0);

GPIO\_ResetBits(GPIOB, GPIO\_Pin\_1);

GPIO\_ResetBits(GPIOF, GPIO\_Pin\_11);

//开始实现系统功能

while (1)

{

/\*\*\*\*\* 根据按键，判断状态 \*\*\*\*\*/

if(KEYPAD\_STATE==keypad\_0)

{

GPIO\_ResetBits(GPIOB, GPIO\_Pin\_14);//关闭电机

Light\_up\_LED(GPIOC, GPIO\_Pin\_4);//自动状态点亮LED

}

if(KEYPAD\_STATE==keypad\_0&&GPIO\_ReadInputDataBit(GPIOC,GPIO\_Pin2))//自动状态下，判断光敏电阻的检测结果

{

GPIO\_SetBits(GPIOB, GPIO\_Pin\_14); //外界过亮开启电机关闭窗帘

printf("Auto light \r");//在终端处显示亮

}

if(KEYPAD\_STATE==keypad\_0&&!GPIO\_ReadInputDataBit(GPIOC,GPIO\_Pin\_2))

{

GPIO\_ResetBits(GPIOB, GPIO\_Pin\_14); //外界过暗

printf("Auto dick \r");

}

if(KEYPAD\_STATE==0x04)//手动状态下无命令

{

GPIO\_SetBits(GPIOC, GPIO\_Pin\_4); //手动状态无指令

printf("Manual \r");

}

if(KEYPAD\_STATE==keypad\_1)//手动状态开启电机

{

GPIO\_SetBits(GPIOC, GPIO\_Pin\_4);

GPIO\_SetBits(GPIOB, GPIO\_Pin\_14);

printf("Manual open \r");

}

if(KEYPAD\_STATE==keypad\_2)//手动状态关闭电机

{

GPIO\_SetBits(GPIOC, GPIO\_Pin\_4);

GPIO\_ResetBits(GPIOB, GPIO\_Pin\_14);

printf("Manual close \r");

}

}

}

系统时钟设置

void RCC\_Configuration(void)

{

/\* RCC system reset(for debug purpose) \*/

RCC\_DeInit();

/\* Enable HSE \*/

RCC\_HSEConfig(RCC\_HSE\_ON);

/\* Wait till HSE is ready \*/

HSEStartUpStatus = RCC\_WaitForHSEStartUp();

if(HSEStartUpStatus == SUCCESS)

{

/\* Enable Prefetch Buffer \*/

FLASH\_PrefetchBufferCmd(FLASH\_PrefetchBuffer\_Enable);

/\* Flash 2 wait state \*/

FLASH\_SetLatency(FLASH\_Latency\_2);

/\* HCLK = SYSCLK \*/

RCC\_HCLKConfig(RCC\_SYSCLK\_Div1);

/\* PCLK2 = HCLK \*/

RCC\_PCLK2Config(RCC\_HCLK\_Div1);

/\* PCLK1 = HCLK/2 \*/

RCC\_PCLK1Config(RCC\_HCLK\_Div2);

/\* ADCCLK = PCLK2/4 \*/

RCC\_ADCCLKConfig(RCC\_PCLK2\_Div4);

/\* PLLCLK = 8MHz \* 9 = 72 MHz \*/

RCC\_PLLConfig(RCC\_PLLSource\_HSE\_Div1, RCC\_PLLMul\_9);

/\* Enable PLL \*/

RCC\_PLLCmd(ENABLE);

/\* Wait till PLL is ready \*/

while(RCC\_GetFlagStatus(RCC\_FLAG\_PLLRDY) == RESET)

{}

/\* Select PLL as system clock source \*/

RCC\_SYSCLKConfig(RCC\_SYSCLKSource\_PLLCLK);

/\* Wait till PLL is used as system clock source \*/

while(RCC\_GetSYSCLKSource() != 0x08)

{

}

}

/\* Enable GPIO and AFIO clocks \*/

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA|RCC\_APB2Periph\_GPIOB|RCC\_APB2Periph\_GPIOC|RCC\_APB2Periph\_GPIOD|RCC\_APB2Periph\_GPIOE|RCC\_APB2Periph\_GPIOF|RCC\_APB2Periph\_GPIOG|RCC\_APB2Periph\_AFIO , ENABLE);

/\* Enable USART1 clocks \*/

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_USART1, ENABLE);

}

中断向量控制器

void NVIC\_Configuration(void)//嵌套中断向量控制器

{

NVIC\_InitTypeDef NVIC\_InitStructure;

#ifdef VECT\_TAB\_RAM

/\* Set the Vector Table base location at 0x20000000 \*/

NVIC\_SetVectorTable(NVIC\_VectTab\_RAM, 0x0);

#else /\* VECT\_TAB\_FLASH \*/

/\* Set the Vector Table base location at 0x08000000 \*/

NVIC\_SetVectorTable(NVIC\_VectTab\_FLASH, 0x0);

#endif

/\* Configure one bit for preemption priority \*/

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_1);//设置优先级分组：先占优先级和从优先级，先占优先级1位，从优先级3位

/\* Enable the EXTI15\_10 Interrupt \*/

NVIC\_InitStructure.NVIC\_IRQChannel = EXTI15\_10\_IRQChannel;//IRQ 通道外部中断线 15-10 中断

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 0;//该参数设置了成员 NVIC\_IRQChannel 中的先占优先级

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 0;//该参数设置了成员 NVIC\_IRQChannel 中的从优先级

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;//开

NVIC\_Init(&NVIC\_InitStructure);

/\* Enable the EXTI9\_5 Interrupt \*/

NVIC\_InitStructure.NVIC\_IRQChannel = EXTI9\_5\_IRQChannel;//外部中断线 9-5 中断

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 0;

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 0;

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;

NVIC\_Init(&NVIC\_InitStructure);

}

外部中断设置

void EXTI\_Configuration(void)//外部中断/事件控制器

{

EXTI\_InitTypeDef EXTI\_InitStructure;

/\* Connect Key Button EXTI Line to Key Button GPIO Pin \*/

GPIO\_EXTILineConfig(GPIO\_PortSourceGPIOC, GPIO\_PinSource13);//选择GPIO管脚用作外部中断线路；PC13引脚的复用功能

GPIO\_EXTILineConfig(GPIO\_PortSourceGPIOG, GPIO\_PinSource15);

GPIO\_EXTILineConfig(GPIO\_PortSourceGPIOF, GPIO\_PinSource7);

/\* Configure Key Button0 EXTI Line to generate an interrupt on falling edge \*/

EXTI\_InitStructure.EXTI\_Line = EXTI\_Line13;//选择了待使能或者失能的外部线路外部中断线 13

EXTI\_InitStructure.EXTI\_Mode = EXTI\_Mode\_Interrupt;//设置了被使能线路的模式，设置 EXTI 线路为中断请求

EXTI\_InitStructure.EXTI\_Trigger = EXTI\_Trigger\_Falling;//设置了被使能线路的触发边沿，设置输入线路下降沿为中断请求

EXTI\_InitStructure.EXTI\_LineCmd = ENABLE;

EXTI\_Init(&EXTI\_InitStructure);

/\* Configure Key Button1 EXTI Line to generate an interrupt on falling edge \*/

EXTI\_InitStructure.EXTI\_Line = EXTI\_Line15;

EXTI\_InitStructure.EXTI\_Mode = EXTI\_Mode\_Interrupt;

EXTI\_InitStructure.EXTI\_Trigger = EXTI\_Trigger\_Falling;

EXTI\_InitStructure.EXTI\_LineCmd = ENABLE;

EXTI\_Init(&EXTI\_InitStructure);

/\* Configure Key Button2 EXTI Line to generate an interrupt on falling edge \*/

EXTI\_InitStructure.EXTI\_Line = EXTI\_Line7;

EXTI\_InitStructure.EXTI\_Mode = EXTI\_Mode\_Interrupt;

EXTI\_InitStructure.EXTI\_Trigger = EXTI\_Trigger\_Falling;

EXTI\_InitStructure.EXTI\_LineCmd = ENABLE;

EXTI\_Init(&EXTI\_InitStructure);

}

void Light\_up\_LED(GPIO\_TypeDef\* GPIOx , u16 GPIO\_Pin)

{

GPIO\_SetBits(GPIO\_LED1,GPIO\_PIN\_LED1); //熄灭LED1

GPIO\_SetBits(GPIO\_LED2,GPIO\_PIN\_LED2);

GPIO\_SetBits(GPIO\_LED3,GPIO\_PIN\_LED3);

GPIO\_SetBits(GPIO\_LED4,GPIO\_PIN\_LED4);

GPIO\_SetBits(GPIO\_LED5,GPIO\_PIN\_LED5);

GPIO\_ResetBits(GPIOx,GPIO\_Pin); //点亮选中的LED

}

GPIO口设置

void GPIO\_Configuration(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

/\* Configure LED1 (PC.4)as alternate function push-pull \*/

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_4;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;

GPIO\_Init(GPIOC, &GPIO\_InitStructure);

/\*GPIOB.14 Configuration:电机 as OutPut push-pull \*/

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_14;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOB, &GPIO\_InitStructure);

/\* Configure 光敏电阻 Pin (PC.2) as input floating \*/

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_2;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;

GPIO\_Init(GPIOC, &GPIO\_InitStructure);

/\* Configure Key Button0 GPIO Pin(PC.13) as input floating (Key Button EXTI Line) \*/

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_13;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;

GPIO\_Init(GPIOC, &GPIO\_InitStructure);

/\* Configure Key Button1 GPIO Pin(PG.15) as input floating (Key Button EXTI Line) \*/

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_15;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;

GPIO\_Init(GPIOG, &GPIO\_InitStructure);

/\* Configure Key Button2 GPIO Pin(PF.7) as input floating (Key Button EXTI Line) \*/

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_7;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;

GPIO\_Init(GPIOF, &GPIO\_InitStructure);

}

延时程序

void Delaytime(u32 nCount)

{

for(; nCount != 0; nCount--);

}

USART串口设置

void USART\_Configuration(void)

{

USART\_InitTypeDef USART\_InitStructure;

USART\_InitStructure.USART\_BaudRate = 115200;//波特率115200

USART\_InitStructure.USART\_WordLength = USART\_WordLength\_8b;//8位数据位

USART\_InitStructure.USART\_StopBits = USART\_StopBits\_1;//1停止位

USART\_InitStructure.USART\_Parity = USART\_Parity\_No ;//无奇偶检验

USART\_InitStructure.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;//无硬件流控制

USART\_InitStructure.USART\_Mode = USART\_Mode\_Rx | USART\_Mode\_Tx;//打开接收发送功能

/\* Configure the USART2 \*/

USART\_Init(USART1 ,&USART\_InitStructure);

/\* Enable the USART2 \*/

USART\_Cmd(USART1, ENABLE);

}

PUTCHAR\_PROTOTYPE

{

/\* Write a character to the USART \*/

USART\_SendData(USART1, (u8) ch);

/\* Loop until the end of transmission \*/

while(USART\_GetFlagStatus(USART1, USART\_FLAG\_TXE) == RESET)

{

}

return ch;

}

#ifdef DEBUG

void assert\_failed(u8\* file, u32 line)

{

/\* User can add his own implementation to report the file name and line number,

ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* Infinite loop \*/

while (1)

{

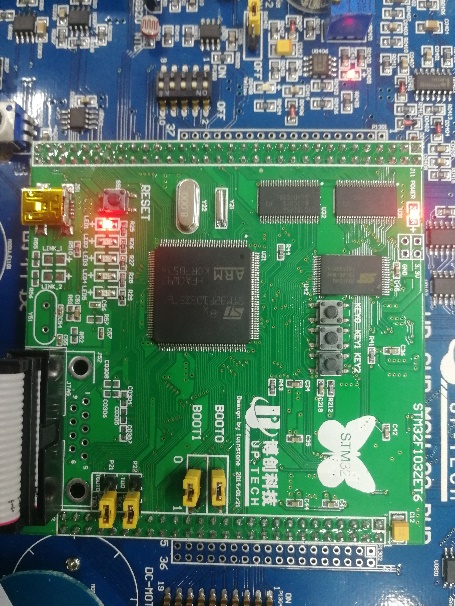
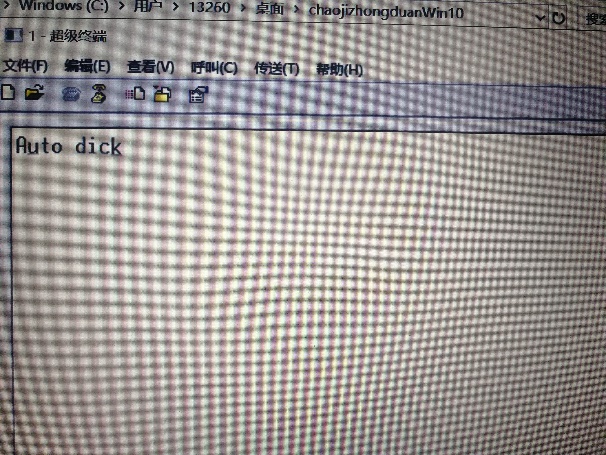
}

}

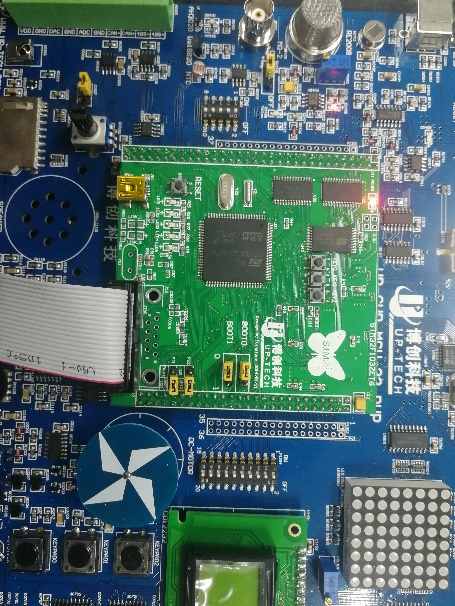
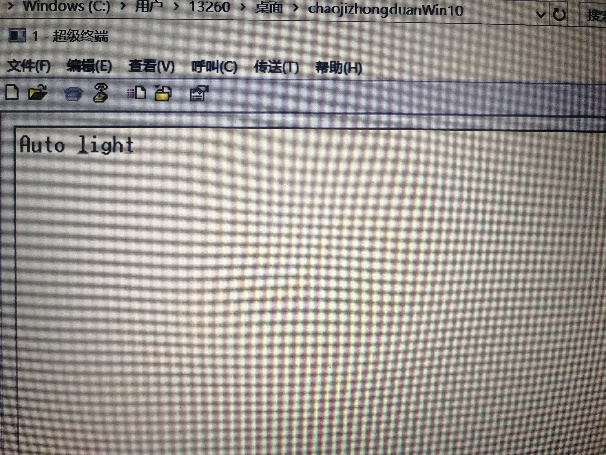
#endif

4．实验结果

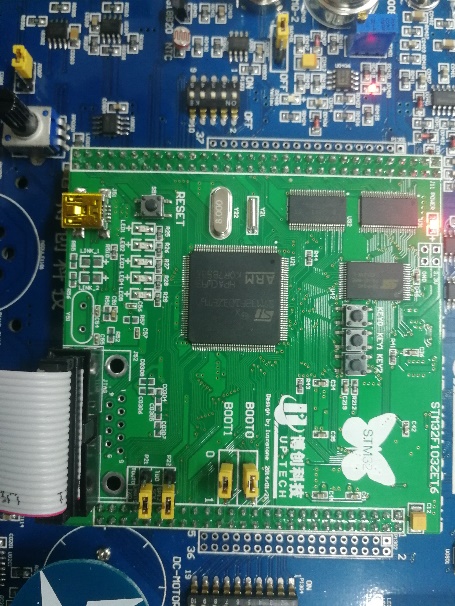
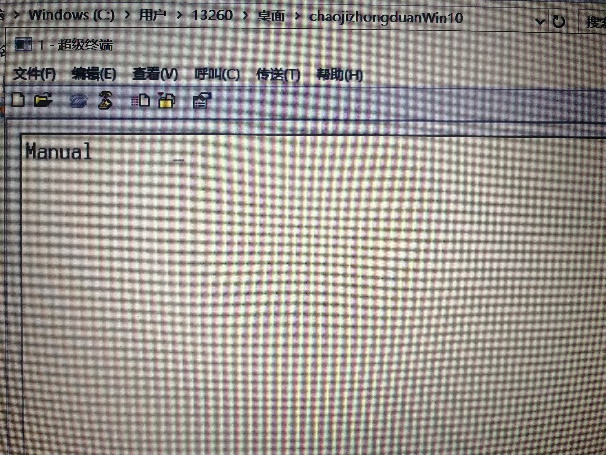
自动状态下光线暗的情况

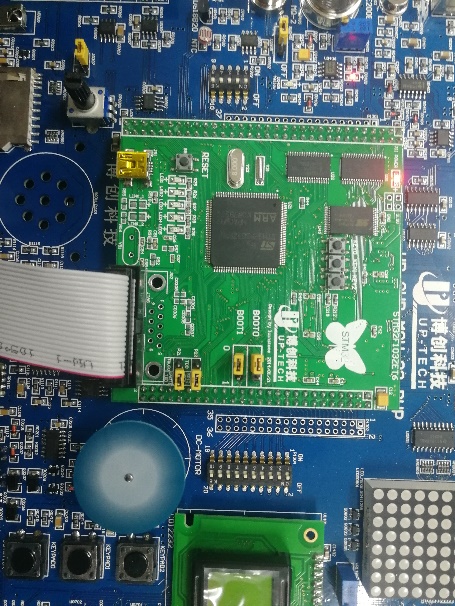
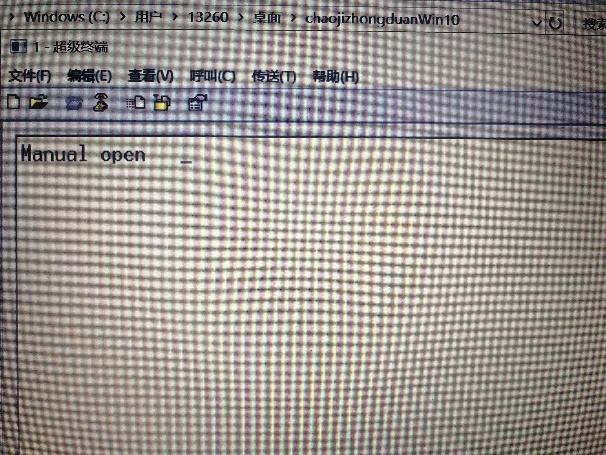
自动状态下光线亮的情况

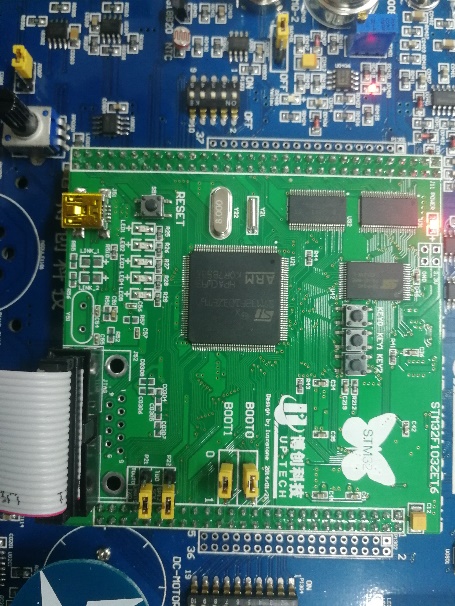
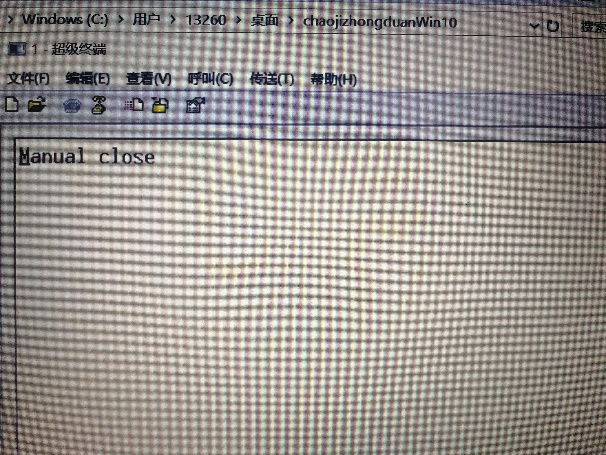
自动状态下无指令状态

自动状态下开启

自动状态下关闭

5．参考文献

1. 田泽.嵌入式系统开发与应用[M]. 北京航空航天大学出版社.2005
2. 田泽.嵌入式系统开发与应用实验教程[M].北京航空航天大学出版社.2004
3. STM32固件库使用手册
4. STM32F103X8-B数据手册（中文）-20150727-CD00161566\_ZHV10