

**毕业设计（论文）附件材料**

基于蓝牙和NB-IOT的数据采集器设计

学生姓名： 罗范支

学 号： 201610213116

所在系部： 电气信息系

专业班级： 16 gb电气3班

指导教师： 李 群 教授（职称）

日 期： 二O二O年5月

# 目 录

目录

[目录 2](#_Toc38803037)

[第2章 程序源码 2](#_Toc38803038)

[2.1 main.c 2](#_Toc38803039)

[2.2 Systerm\_Init.c 4](#_Toc38803040)

[2.3 NB.c 7](#_Toc38803041)

[2.4 led.c 24](#_Toc38803042)

[2.5 key.c 26](#_Toc38803043)

[2.6 NB\_time.c 29](#_Toc38803044)

[2.7 wdg.c 32](#_Toc38803045)

[2.8 BlueTooth.c 33](#_Toc38803046)

[2.9 SimplePrintf.c 42](#_Toc38803047)

[2.10 Blue\_time.c 50](#_Toc38803048)

[2.11 BlueData\_Deal.c 54](#_Toc38803049)

[2.12 Communication\_to\_app.c 83](#_Toc38803050)

[2.13 Oled.c 96](#_Toc38803051)

[2.14 Beep.c 109](#_Toc38803052)

[2.15 Flash.c 110](#_Toc38803053)

[系统原理图设计如所示 115](#_Toc38803054)

[第3章 系统PCB板 116](#_Toc38803055)

[第4章 PCB 3D图 117](#_Toc38803056)

[第5章 实物图 118](#_Toc38803057)

# 图表目录

[图 2‑1 图系统原理图 116](#_Toc38803898)

[图 3‑1 系统PCB板图 117](#_Toc38803899)

[图 4‑1数据采集器PCB 3D图 118](#_Toc38803900)

[图 5‑1数据采集器实物图 119](#_Toc38803901)

# 程序源码

## main.c

#include "Systerm\_Init.h"

#include "stm32f10x\_it.h"

#include "oled.h"

#include "stdlib.h"

#include "delay.h"

#include "beep.h"

#define Scan\_min(n) 60000\*n //定时扫描事件

int main(void)

{

SystemInit(); //系统时钟初始化

SYSTERM\_INIT(); //系统初始化

OLED\_Init(); //OLED初始化

OLED\_ColorTurn(0); //0正常显示，1- 反色显示

OLED\_DisplayTurn(0); //0正常显示，1-屏幕翻转显示

OLED\_Clear(); //清屏

BLUE\_POWER\_OPEN(); //打开蓝牙电源

Deveice\_Display(); //设备信息显示

BlueConnectStruct.Connect=0;

while(1)

{

NB\_INIT(); //上电初始化NB，储存EMI号和信号强度

KEY\_Set\_Blue\_S\_Mode(); //短按按键蓝牙切换成从模式和APP通讯

APP\_To\_BlueScan(Scan\_min(5)); //APP启动扫描, 计时扫描单位：分钟

Determine\_Deveice(); //APP通讯错误信息返回

JZQ\_Write\_NAME\_MAC(); //APP写表号

NB\_UPDATE\_DATA(); //NB上传表数据--最大上传40个蓝牙水表

//Wake\_Deal(); //定时唤醒处理

if(BlueConnectStruct.Connect==1)

{

if(BLUE\_CONNECT\_COND==0)

{

OLED\_Display\_GB2312\_string(1,6,"蓝牙等待连接...");

}

if(BLUE\_CONNECT\_COND==1)

{

OLED\_Display\_GB2312\_string(1,6,"蓝牙已连接！！！");

BlueConnectStruct.Connect=2;

}

}

else if(BlueConnectStruct.Connect==2)

{

if(BLUE\_CONNECT\_COND==0)

{

BlueConnectStruct.Connect=1;

}

}

}

}

## Systerm\_Init.c

#include "Systerm\_Init.h"

#include "beep.h"

#include "oled.h"

void SYSTERM\_INIT(void)

{

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2); //中断优先级分组

USART1\_Init(9600); //NB的串口

USART2\_Init(115200); //蓝牙串口

USART3\_Init(115200); //DEBUG串口

TIM3\_Init(99,4799);//10K的计数频率，重装值为100

TIM4\_Init(99,479);//1ms

delay\_init();

LED\_GPIO\_Init();

KEY\_GPIO\_Init();

NB\_GPIO\_Init();

BLUE\_GPIO\_Init();

LED\_Init();

NB\_ALL\_Flag\_Init();

//IWDG\_Init(6,28125);//独立看门狗初始化-预分频系数为64，重载值为625，溢出时间为1S //(4\*2^5)\*31250/40 ms

RTC\_Init();

RTC\_NVIC\_Config();

ConnectBlueStruct.Connect\_quality=20;//连接表端的结点数

Read\_NAME\_AND\_NAME();//读取缓存的表号和MAC地址

BEEP\_GPIO\_Init();

PrintfFlag=1;

ReceivMAC\_Flag=0;

}

void LED\_Init(void)

{

LEDn\_ON(4);

delay\_ms(1000);

LEDn\_OFF(4);

}

void NB\_ALL\_Flag\_Init(void)

{

NB\_Flag\_Init();

Blue\_Flag\_Init();

BlueConnectStruct.BlueConnect\_time=0;

BlueDataRceiveModeStruct.DataMode.Start\_NB=0;

NB\_AT\_Num=0;

BlueDataRceiveModeStruct.Blue\_AT\_Mode=0;

BlueDataRceiveModeStruct.Blue\_Data\_Mode=0;

ConnectBlueStruct.Connect\_Number=0;

ConnectBlueStruct.ReadBlueData\_Success=0;

BlueDataRceiveModeStruct.DataMode.BlueDataCount=0;

ConnectBlueStruct.StopScan=0;

ConnectBlueStruct.Connect\_OverTime\_Flag=0;

ConnectBlueStruct.ReadBlueData\_fail=0;

FLASH\_Struct.Write\_Success=0;

k=0;

Sleep\_Mode\_Struct.Enter\_SleepMode\_Flag=0;

Sleep\_Mode\_Struct.WeakUP=0;

BlueDataRceiveModeStruct.Blue\_To\_APP=0;

CommunicationStruct.Communication\_RX\_Cont=0;

CommunicationStruct.APP\_To\_JQZ\_CB\_Flag=0;

NB\_Struct.NB\_Init=0;

Work\_Struct.Mode\_type.JQZ\_Mode.Update=0;

Work\_Struct.Mode\_type.Weakup.Set\_ARM\_Time=0;

Work\_Struct.Mode\_type.Weakup.WeakUP\_Over\_Time=0;

Work\_Struct.Mode\_type.Sleep\_Mode=0;

Work\_Struct.Mode\_type.Blue\_S.Con\_over\_Time=0;

CommunicationStruct.APP\_To\_JQZ\_CB\_Flag=0;

CommunicationStruct.JQZ\_CB\_UPDATE=0;

NB\_Struct.NB\_1=1;

NB\_Struct.Net\_time=0;

}

void Deveice\_Display(void)

{

OLED\_Display\_GB2312\_string(1,0,"湖北工业大学");

OLED\_Display\_GB2312\_string(1,2,"工程技术学院");

OLED\_Display\_GB2312\_string(1,4,"姓名：罗范支");

OLED\_Display\_GB2312\_string(1,6,"学号201610213116");

delay\_s(2);

OLED\_Clear();

OLED\_Display\_GB2312\_string(1,0,"毕业设计题目：");

OLED\_Display\_GB2312\_string(1,2,"基于蓝牙和NB模组");

OLED\_Display\_GB2312\_string(1,4,"的数据收集器");

OLED\_Display\_GB2312\_string(1,6,"指导老师：李群");

delay\_s(2);

OLED\_Clear();

OLED\_Display\_GB2312\_string(1,0,"系统初始化...");

BEEP\_ON(1);

}

## NB.c

#include "NB.h"

#include "usart.h"

#include "string.h"

#include "led.h"

#include "BlueData\_Deal.h"

#include "SimplePrintf.h"

#include "delay.h"

#include "communication\_to\_app.h"

#include "flash.h"

#include "wakeup.h"

#include "oled.h"

#include "beep.h"

u16 NB\_time;

u8 NB\_SendData\_flag;

u8 NB\_ReceveData\_flag;

u8 NB\_AT\_Num;

u8 NB\_ReceveData\_Failed\_Flag;

u16 NB\_Close\_Power\_time;

NB\_Typedef NB\_Struct;

char Device\_information\_char\_Buff[50];

uint8\_t Device\_information\_u8\_Buff[50];

//NB模组IO口初始化

void NB\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

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

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

GPIO\_InitStructure.GPIO\_Pin =GPIO\_Pin\_8|GPIO\_Pin\_12;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_2MHz;

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

GPIO\_ResetBits(GPIOA,GPIO\_Pin\_8|GPIO\_Pin\_12);

//GPIO\_SetBits

//GPIO\_ResetBits

}

void NB\_NO\_Rest()

{

NB\_Switch=0;

}

void NB\_Rest()

{

NB\_Switch=1;

}

#include "time.h"

u8 CurrentTim[6];//年，月，日，时，分。各一个字节

struct tm GTM\_Time;

void GTM\_ConversionUTC(uint16\_t Count)

{

static u32 TimSec;

GTM\_Time.tm\_year = 2000+CurrentTim[0]-1900;//年

GTM\_Time.tm\_mon = CurrentTim[1]-1;//月

GTM\_Time.tm\_mday = CurrentTim[2];//日

GTM\_Time.tm\_hour = CurrentTim[3];//时

GTM\_Time.tm\_min = CurrentTim[4];//分

GTM\_Time.tm\_sec = CurrentTim[5];//秒

TimSec = mktime(&GTM\_Time);

TimSec += Count;

localtime\_r(&TimSec, &GTM\_Time);

CurrentTim[0] = GTM\_Time.tm\_year+1900-2000 ;//年

CurrentTim[1]= GTM\_Time.tm\_mon+1 ;//月

CurrentTim[2]= GTM\_Time.tm\_mday ;//日

CurrentTim[3] = GTM\_Time.tm\_hour ;//时

CurrentTim[4]= GTM\_Time.tm\_min ;//分

CurrentTim[5] = GTM\_Time.tm\_sec ;//秒

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*NB上传数据函数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void NB\_Update\_Data(void)

{

char cmd[129];

strcpy(cmd, "AT+NMGS=59,");

strcat(cmd,BlueDataRceiveModeStruct.DataMode.BlueDataStringBuff);

memset(BlueDataRceiveModeStruct.DataMode.BlueDataStringBuff,0,sizeof(BlueDataRceiveModeStruct.DataMode.BlueDataStringBuff));

memset(BlueDataRceiveModeStruct.DataMode.BlueDataStringBuff,0,sizeof(BlueDataRceiveModeStruct.DataMode.BlueDataStringBuff));

Uart1\_SendAtCommand(cmd);

}

void NB\_Flag\_Init(void)

{

NB\_AT\_Num =0;

NB\_SendData\_flag =1;

NB\_ReceveData\_flag=1;

NB\_Close\_Power\_time=0;

}

uint8\_t NB\_Bard\_AT\_Send(u8 a)

{

uint8\_t m,n;

static char temp[12];

static char i;

static char IP[40];

STMFLASH\_Read(FLASH\_SAVE\_ADDR+2048,(u16\*)CommunicationStruct.IP\_COM\_u8,20);

HEXArrayToStringArray((u8\*)CommunicationStruct.IP\_COM\_u8,CommunicationStruct.IP\_COM\_char,20);

StringArrayToHEXArray(CommunicationStruct.IP\_COM\_char,CommunicationStruct.IP\_COM\_string,40);

strcpy(IP,"AT+NCDP=");

for(i=0;i<15;i++){IP[i+8]=CommunicationStruct.IP\_COM\_string[i+5];}

IP[23]=CommunicationStruct.IP\_COM\_string[4];

IP[24]=CommunicationStruct.IP\_COM\_string[0];

IP[25]=CommunicationStruct.IP\_COM\_string[1];

IP[26]=CommunicationStruct.IP\_COM\_string[2];

IP[27]=CommunicationStruct.IP\_COM\_string[3];

if(NB\_SendData\_flag)

{

NB\_SendData\_flag=0;

switch(NB\_AT\_Num)

{

case 0: NB\_Power\_Control=1;// 1-开启NB电源，0-关闭NB电源

//NB\_Rest();

NB\_time=50;

break;

case 1: USart\_CLR\_RecvBuf(1);

Uart1\_SendAtCommand("AT");//测试AT命令

NB\_time=50;

break;

case 2:

if(strstr(USART1\_RX\_Buff,"")!=NULL)

{

USart\_CLR\_RecvBuf(1);

NB\_ReceveData\_Failed\_Flag=0;

Uart1\_SendAtCommand("AT+NCDP=180.101.147.115,5683");//设置IoT平台IP地址

//Uart1\_SendAtCommand(IP);//设置IoT平台IP地址

NB\_time=50;

}

break;

case 3:

if(strstr(USART1\_RX\_Buff,"")!=NULL)

{

USart\_CLR\_RecvBuf(1);

Uart1\_SendAtCommand("AT+CGMR");//查询固件版本

NB\_time=50;

}

break;

case 4: if(strstr(USART1\_RX\_Buff,"")!=NULL)//SECURITY

{

USart\_CLR\_RecvBuf(1);

Uart1\_SendAtCommand("AT+CGSN=1");//查询IMEI号

NB\_time=100;

}

break;

case 5: if(strstr(USART1\_RX\_Buff,"+CGSN:")!=NULL)

{

for(m=0;m<50;m++)

{

if(USART1\_RX\_Buff[m-2]=='N'&&USART1\_RX\_Buff[m-1]==':')

{

for(n=0;n<15;n++){Device\_information\_char\_Buff[n]=USART1\_RX\_Buff[n+m];}

Device\_information\_char\_Buff[15]='0';

}

}

USart\_CLR\_RecvBuf(1);

Uart1\_SendAtCommand("AT+NCONFIG?");//配置自动联网，扰码开启

NB\_time=50;

}

else

{

USart\_CLR\_RecvBuf(1);

Uart1\_SendAtCommand("AT+CGSN=1");//查询IMEI号

NB\_AT\_Num=4;

}

break;

case 6: if(strstr(USART1\_RX\_Buff,"+NCONFIG:AUTOCONNECT,TRUE")!=NULL)

{

USart\_CLR\_RecvBuf(1);

Uart1\_SendAtCommand("AT+NBAND?");

NB\_time=50;

}

case 7: if(strstr(USART1\_RX\_Buff, "+NBAND:5") != NULL)

{

NB\_ReceveData\_Failed\_Flag=0;

USart\_CLR\_RecvBuf(1);

Uart1\_SendAtCommand("AT+CFUN=1");

NB\_time=50;

}

break;

case 8: if(strstr(USART1\_RX\_Buff,"OK")!=NULL)

{

NB\_ReceveData\_Failed\_Flag=0;

USart\_CLR\_RecvBuf(1);

Uart1\_SendAtCommand("AT+CGATT?");

NB\_time=50;

}

break;

//

case 9: if(strstr(USART1\_RX\_Buff,"+CGATT:1")!=NULL)

{

NB\_ReceveData\_Failed\_Flag=0;

NB\_ReceveData\_Failed\_Flag=0;

USart\_CLR\_RecvBuf(1);

Uart1\_SendAtCommand("AT+CIMI");

NB\_time=50;

}

else

{

USart\_CLR\_RecvBuf(1);

Uart1\_SendAtCommand("AT+CGATT?");

NB\_AT\_Num=8;

NB\_ReceveData\_Failed\_Flag++;

if(NB\_ReceveData\_Failed\_Flag>30)

{

if(NB\_Struct.Net\_time==1)

{

OLED\_Display\_GB2312\_string(1,0,"SIM卡驻网失败！");

OLED\_Display\_GB2312\_string(1,2,"请检查SIM卡！");

OLED\_Display\_GB2312\_string(1,4,"确认无误后请重启");

NB\_AT\_Num=20;

break;

}

if(NB\_Struct.Net\_time<1)

{

NB\_Struct.Net\_time+=1;

NB\_Flag\_Init();

NB\_Power\_Control=0;

delay\_ms(500);

NB\_AT\_Num=0;

}

// Work\_Struct.Mode\_type.Sleep\_Mode=1;

// USart\_CLR\_RecvBuf(1);

// Uart1\_SendAtCommand("AT+CIMI");

// NB\_AT\_Num=9;

NB\_ReceveData\_Failed\_Flag=0;

}

}

break;

//

case 10:

if(strstr(USART1\_RX\_Buff,"460")!=NULL)

{

NB\_ReceveData\_Failed\_Flag=0;

USart\_CLR\_RecvBuf(1);

Uart1\_SendAtCommand("AT+NCCID");

NB\_time=20;

}

break;

//

case 11: if(strstr(USART1\_RX\_Buff,"+NCCID")!=NULL)

{

USart\_CLR\_RecvBuf(1);

Uart1\_SendAtCommand("AT+CSQ");

NB\_time=50;

}

break;

//

case 12: if(strstr(USART1\_RX\_Buff,"+CSQ:")!=NULL)

{

for(m=0;m<20;m++)

{

if(USART1\_RX\_Buff[m-2]=='Q'&&USART1\_RX\_Buff[m-1]==':')

{

if(USART1\_RX\_Buff[m+1]!=',')

{

Device\_information\_char\_Buff[16]=USART1\_RX\_Buff[m];

Device\_information\_char\_Buff[17]=USART1\_RX\_Buff[m+1];

}

else if(USART1\_RX\_Buff[m+1]==',')

{

Device\_information\_char\_Buff[16]='0';

Device\_information\_char\_Buff[17]=USART1\_RX\_Buff[m];

}

}

}

USart\_CLR\_RecvBuf(1);

Uart1\_SendAtCommand("AT+CEREG?");

NB\_time=5;

}

break;

//

case 13: if(strstr(USART1\_RX\_Buff, "+CEREG:0,1") != NULL)

{

USart\_CLR\_RecvBuf(1);

Uart1\_SendAtCommand("AT+CSCON?");

NB\_ReceveData\_Failed\_Flag=0;

NB\_time=50;

}

else

{

NB\_AT\_Num=12;

NB\_ReceveData\_Failed\_Flag++;

Uart1\_SendAtCommand("AT+CEREG?");

NB\_time=50;

}

break;

//

case 14: if(strstr(USART1\_RX\_Buff, "+CSCON:0,1") != NULL)

{

NB\_ReceveData\_Failed\_Flag=0;

USart\_CLR\_RecvBuf(1);

Uart1\_SendAtCommand("AT+CCLK?");

NB\_time=50;

}

else

{

NB\_AT\_Num=13;

USart\_CLR\_RecvBuf(1);

Uart1\_SendAtCommand("AT+CSCON?");

NB\_time=50;

NB\_ReceveData\_Failed\_Flag++;

if(NB\_ReceveData\_Failed\_Flag>100)

{

Work\_Struct.Mode\_type.Sleep\_Mode=1;

NB\_ReceveData\_Failed\_Flag=0;

}

}

break;

case 15: if(strstr(USART1\_RX\_Buff, "+CCLK:") != NULL)

{

temp[0]=USART1\_RX\_Buff[8];//年

temp[1]=USART1\_RX\_Buff[9];

temp[2]=USART1\_RX\_Buff[11];//月

temp[3]=USART1\_RX\_Buff[12];

temp[4]=USART1\_RX\_Buff[14];//日

temp[5]=USART1\_RX\_Buff[15];

temp[6]=USART1\_RX\_Buff[17];//时

temp[7]=USART1\_RX\_Buff[18];

temp[8]=USART1\_RX\_Buff[20];//分

temp[9]=USART1\_RX\_Buff[21];

temp[10]=USART1\_RX\_Buff[23];//秒

temp[11]=USART1\_RX\_Buff[24];

AscillToString(temp, temp,12);//Ascii转16进制

for(i=0;i<6;i++)

CurrentTim[i] = temp[i\*2]\*10+temp[i\*2+1];//数组高低位合并

GTM\_ConversionUTC(28800);//3600\*8小时

USart\_CLR\_RecvBuf(1);

Uart1\_SendAtCommand("AT+NNMI=0");

NB\_time=100;

}

break;

case 16: if(strstr(USART1\_RX\_Buff, "OK") != NULL)

{

if(a==1)

{

STMFLASH\_Write(FLASH\_SAVE\_ADDR+4096,(u16\*)Device\_information\_char\_Buff,18);

NB\_Struct.NB\_Init=1;

ConnectBlueStruct.Read\_Flag=0;

USart\_CLR\_RecvBuf(1);

NB\_time=100;

USART3\_Send\_Arr(Device\_information\_char\_Buff,18);

OLED\_Clear();

OLED\_Display\_GB2312\_string(1,0,"系统初始化成功!!");

OLED\_Display\_GB2312\_string(1,4,"长按按键启动蓝牙");

BEEP\_ON(3);

}

else if(a==2)

{

STMFLASH\_Write(FLASH\_SAVE\_ADDR+4096,(u16\*)Device\_information\_char\_Buff,18);

ConnectBlueStruct.Read\_Flag=0;

ConnectBlueStruct.BlueScanSuccess=1;

ConnectBlueStruct.connect\_flag=1;

CommunicationStruct.APP\_To\_JQZ\_CB\_Flag=0;

USart\_CLR\_RecvBuf(1);

NB\_time=100;

}

//NB\_Update\_Data();

}

break;

default : NB\_time=0;

NB\_AT\_Num=0;

NB\_SendData\_flag=0;

NB\_ReceveData\_flag=0;

break;

}

}

return NB\_ReceveData\_Failed\_Flag;

}

void NB\_Iot\_AT\_Send(u8 b)

{

if(NB\_SendData\_flag)

{

if(NB\_ReceveData\_flag)//

{

NB\_ReceveData\_flag=0;

NB\_Bard\_AT\_Send(b);

}

}

}

void NB\_INIT(void)

{

if(ConnectBlueStruct.Read\_Flag)//启动NB

{

NB\_Struct.Mode =1;

NB\_Iot\_AT\_Send(1);

}

}

void NB\_UPDATE\_DATA(void)//NB上传数据

{

u8 i;

char name[142];

char mac[122];

if(CommunicationStruct.APP\_To\_JQZ\_CB\_Flag==0x01)

{

NB\_Struct.Mode=1;

Uart2\_SendAtCommand("AT+DISCONNECT");

delay\_ms(200);

Uart2\_SendAtCommand("AT+ROLE=M");

STMFLASH\_Read(SAVE\_ADDR\_NAME\_1,(u16\*)name,70);

STMFLASH\_Read(SAVE\_ADDR\_MAC\_1,(u16\*)mac,60);

for(i=0;i<140;i++){BlueNAME[i]=name[i];}

for(i=0;i<120;i++){BlueMAC[i]=mac[i];}

ConnectBlueStruct.Connect\_lenth =strlen(BlueNAME)/14\*2;

StringToBytes(BlueNAME,StringConvert16\_BlueNAME,140);//蓝牙名称字符串转16进制

NB\_Flag\_Init();

NB\_Power\_Control=0;

delay\_ms(1000);

NB\_Power\_Control=1;

CommunicationStruct.APP\_To\_JQZ\_CB\_Flag++;

}

if(CommunicationStruct.JQZ\_CB\_UPDATE==0x01&&CommunicationStruct.APP\_To\_JQZ\_CB\_Flag==0x02)

{

//LED3=!LED3;

NB\_Struct.Mode =1;

NB\_Iot\_AT\_Send(2);

}

if(Work\_Struct.Mode\_type.JQZ\_Mode.Update==1)//第二次上传10个

{

STMFLASH\_Read(SAVE\_ADDR\_NAME\_2,(u16\*)name,70);

STMFLASH\_Read(SAVE\_ADDR\_MAC\_2,(u16\*)mac,60);

for(i=0;i<140;i++){BlueNAME[i]=name[i];}

for(i=0;i<120;i++){BlueMAC[i]=mac[i];}

ConnectBlueStruct.Connect\_lenth =strlen(BlueNAME)/14\*2;

if(ConnectBlueStruct.Connect\_lenth<10||(BlueMAC[60]=='0'&&BlueMAC[61]=='0'&&BlueMAC[62]=='0'&&BlueMAC[63]=='0'))

{

ConnectBlueStruct.BlueScanSuccess=0;

ConnectBlueStruct.connect\_flag=0;

delay\_ms(100);

raw\_printf("抄表结束");

Work\_Struct.Mode\_type.JQZ\_Mode.Update=0;

}

else

{

StringToBytes(BlueNAME,StringConvert16\_BlueNAME,140);//蓝牙名称字符串转16进制

ConnectBlueStruct.BlueScanSuccess=1;

ConnectBlueStruct.connect\_flag=1;

Work\_Struct.Mode\_type.JQZ\_Mode.Update++;

}

}

if(Work\_Struct.Mode\_type.JQZ\_Mode.Update==3)//第三次上传10个

{

STMFLASH\_Read(SAVE\_ADDR\_NAME\_3,(u16\*)name,70);

STMFLASH\_Read(SAVE\_ADDR\_MAC\_3,(u16\*)mac,60);

for(i=0;i<140;i++){BlueNAME[i]=name[i];}

for(i=0;i<120;i++){BlueMAC[i]=mac[i];}

ConnectBlueStruct.Connect\_lenth =strlen(BlueNAME)/14\*2;

if(ConnectBlueStruct.Connect\_lenth<10||(BlueMAC[60]=='0'&&BlueMAC[61]=='0'&&BlueMAC[62]=='0'&&BlueMAC[63]=='0'))

{

ConnectBlueStruct.BlueScanSuccess=0;

ConnectBlueStruct.connect\_flag=0;

delay\_ms(100);

raw\_printf("抄表结束");

Work\_Struct.Mode\_type.JQZ\_Mode.Update=0;

}

else

{

StringToBytes(BlueNAME,StringConvert16\_BlueNAME,140);//蓝牙名称字符串转16进制

ConnectBlueStruct.BlueScanSuccess=1;

ConnectBlueStruct.connect\_flag=1;

Work\_Struct.Mode\_type.JQZ\_Mode.Update++;

}

}

if(Work\_Struct.Mode\_type.JQZ\_Mode.Update==5)//第四次上传10个

{

STMFLASH\_Read(SAVE\_ADDR\_NAME\_4,(u16\*)name,70);

STMFLASH\_Read(SAVE\_ADDR\_MAC\_4,(u16\*)mac,60);

for(i=0;i<140;i++){BlueNAME[i]=name[i];}

for(i=0;i<120;i++){BlueMAC[i]=mac[i];}

ConnectBlueStruct.Connect\_lenth =strlen(BlueNAME)/14\*2;//实际表节点数

if(ConnectBlueStruct.Connect\_lenth<10||(BlueMAC[60]=='0'&&BlueMAC[61]=='0'&&BlueMAC[62]=='0'&&BlueMAC[63]=='0'))

{

ConnectBlueStruct.BlueScanSuccess=0;

ConnectBlueStruct.connect\_flag=0;

delay\_ms(100);

raw\_printf("抄表结束");

Work\_Struct.Mode\_type.JQZ\_Mode.Update=0;

}

else

{

StringToBytes(BlueNAME,StringConvert16\_BlueNAME,140);//蓝牙名称字符串转16进制

ConnectBlueStruct.BlueScanSuccess=1;

ConnectBlueStruct.connect\_flag=1;

Work\_Struct.Mode\_type.JQZ\_Mode.Update++;

}

}

if(Work\_Struct.Mode\_type.JQZ\_Mode.Update==7)

{

ConnectBlueStruct.BlueScanSuccess=0;

ConnectBlueStruct.connect\_flag=0;

delay\_ms(100);

raw\_printf("抄表结束");

Work\_Struct.Mode\_type.JQZ\_Mode.Update=0;

}

if(ConnectBlueStruct.connect\_flag)//读表数据并上传

{

BLUE\_POWER\_OPEN();

BLUE\_EN\_ENABLE();

Connect\_Blue\_Read\_Data(ConnectBlueStruct.Connect\_quality);

Connect\_Fail\_Deal();//第一次读表数据失败的进行第二次读表数据

}

}

## led.c

#include "led.h"

#include "sys.h"

void LED\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

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

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

GPIO\_InitStructure.GPIO\_Pin =GPIO\_Pin\_12|GPIO\_Pin\_13|GPIO\_Pin\_14;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_2MHz;

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

GPIO\_SetBits(GPIOB,GPIO\_Pin\_12|GPIO\_Pin\_13|GPIO\_Pin\_14);

}

void LEDn\_ON(u8 num)

{

switch(num)

{

case 1:LED1=0;

break;

case 2:LED2=0;

break;

case 3:LED3=0;

break;

case 4:LED1=0;LED2=0;LED3=0;

break;

default:

break;

}

}

void LEDn\_OFF(u8 num)

{

switch(num)

{

case 1:LED1=1;

break;

case 2:LED2=1;

break;

case 3:LED3=1;

break;

case 4:LED1=1;LED2=1;LED3=1;

break;

default:

break;

}

}

## key.c

#include "key.h"

#include "sys.h"

#include "delay.h"

#include "BlueData\_Deal.h"

#include "usart.h"

#include "led.h"

#include "BlueTooth.h"

#define StartScan 1

KEYTypedef KEYStruct;

void KEY\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE); // led4

GPIO\_InitStructure.GPIO\_Pin =GPIO\_Pin\_0; //

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPU; //

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_2MHz; //

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

}

u8 KEY\_Scan(u8 mode)//mode-1-支持连按

{

static u8 key\_up=1;//按键松开标志

if(mode)key\_up=1;//支持连按

if(key\_up&&(KEY1==0||KEY2==0||KEY3==0))

{

delay\_ms(5);

key\_up=0;

if(KEY1==0)return 1;

else if(KEY2==0)return 2;

else if(KEY3==0)return 3;

}

else if(KEY1==1&&KEY2==1&&KEY3==1)key\_up=1;

return 0;//无按键按下

}

void KEY\_SCAN(uint8\_t Jishi\_Time)//计时单位为：分钟

{

if(KEY\_Scan(1)==2)

{

KEYStruct.KEY\_MODE.short\_key=1;

KEYStruct.KEY\_Time++;

if(KEYStruct.KEY\_Time>50)

{

BLUE\_POWER\_OPEN();

BLUE\_EN\_ENABLE();

BlueDataRceiveModeStruct.Blue\_AT\_Mode=2;

ConnectBlueStruct.ConnectBlueReceive\_AT=1;;//串口接收切换成扫描模式

ConnectBlueStruct.ScanTime=Jishi\_Time\*60000;

KEYStruct.KEY\_MODE.long\_key=1;

Uart2\_SendAtCommand("AT+ROLE=M");

delay\_ms(200);

Uart2\_SendAtCommand("AT+SCAN:RSSI=-80");

KEYStruct.KEY\_Time=600;

}

else

{

KEYStruct.KEY\_MODE.short\_key=1;

}

}

if(StartScan==1&&KEYStruct.KEY\_MODE.long\_key==1)

{

LED1=0;

Take\_Out\_NAME\_MAC(80);

if(ConnectBlueStruct.StopScan)

{

KEYStruct.KEY2\_Flag=0;

ConnectBlueStruct.StopScan=0;

USart\_CLR\_RecvBuf(2);

#undef StartScan

}

}

}

## NB\_time.c

#include "NB\_time.h"

#include "sys.h"

#include "NB.h"

#include "led.h"

#include "BlueTooth.h"

#include "BlueData\_Deal.h"

#include "Blue\_time.h"

#include "BlueTooth.h"

//通用定时器3中断初始化

//这里时钟选择为APB1的2倍，而APB1为36M

//arr：自动重装值。

//psc：时钟预分频数

//这里使用的是定时器3!

void TIM3\_Init(u16 arr,u16 psc)

{

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;

NVIC\_InitTypeDef NVIC\_InitStructure;

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM3, ENABLE); //时钟使能

//定时器TIM3初始化

TIM\_TimeBaseStructure.TIM\_Period = arr; //设置在下一个更新事件装入活动的自动重装载寄存器周期的值

TIM\_TimeBaseStructure.TIM\_Prescaler =psc; //设置用来作为TIMx时钟频率除数的预分频值

TIM\_TimeBaseStructure.TIM\_ClockDivision = TIM\_CKD\_DIV1; //设置时钟分割:TDTS = Tck\_tim

TIM\_TimeBaseStructure.TIM\_CounterMode = TIM\_CounterMode\_Down; //TIM向下计数模式

TIM\_TimeBaseInit(TIM3, &TIM\_TimeBaseStructure); //根据指定的参数初始化TIMx的时间基数单位

TIM\_ITConfig(TIM3,TIM\_IT\_Update,ENABLE ); //使能指定的TIM3中断,允许更新中断

//中断优先级NVIC设置

NVIC\_InitStructure.NVIC\_IRQChannel = TIM3\_IRQn; //TIM3中断

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 0; //先占优先级0级

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 2; //从优先级3级

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE; //IRQ通道被使能

NVIC\_Init(&NVIC\_InitStructure); //初始化NVIC寄存器

TIM\_Cmd(TIM3, ENABLE); //使能TIMx

}

//定时器3中断服务程序

void TIM3\_IRQHandler(void) //TIM3中断

{

if (TIM\_GetITStatus(TIM3, TIM\_IT\_Update) != RESET) //检查TIM3更新中断发生与否

{

TIM\_ClearITPendingBit(TIM3, TIM\_IT\_Update ); //清除TIMx更新中断标志

if(NB\_time>0)

{

NB\_time--;

if(NB\_time==0)

{

NB\_AT\_Num++;

//LED1=!LED1;

NB\_SendData\_flag=1;

NB\_ReceveData\_flag=1;

}

}

if(NB\_Close\_Power\_time>0)

{

NB\_Close\_Power\_time--;

if(NB\_Close\_Power\_time==0)

{

NB\_Power\_Control=0;

LEDn\_OFF(1);

LEDn\_OFF(2);

LEDn\_OFF(3);

}

}

if(Blue\_time >0)

{

Blue\_time--;

if(Blue\_time ==0)

{

Blue\_AT\_num ++;

Blue\_ReceveData\_Flag=1;

Blue\_SendData\_Flag =1;

}

}

if(ConnectBlueStruct.Connect\_Time>0)

{

ConnectBlueStruct.Connect\_Time--;

if(ConnectBlueStruct.Connect\_Time==0)

{

ConnectBlueStruct.Connect\_OverTime\_Flag=1;

}

}

}

}

## wdg.c

#include "wdg.h"

//初始化独立看门狗

//prer:分频数:0~7(只有低3位有效!)

//分频因子=4\*2^prer.但最大值只能是256!

//rlr:重装载寄存器值:低11位有效.

//时间计算(大概):Tout=((4\*2^prer)\*rlr)/40 (ms).

void IWDG\_Init(u8 prer,u16 rlr)

{

IWDG\_WriteAccessCmd(IWDG\_WriteAccess\_Enable); //使能对寄存器IWDG\_PR和IWDG\_RLR的写操作

IWDG\_SetPrescaler(prer); //设置IWDG预分频值:设置IWDG预分频值为64

IWDG\_SetReload(rlr); //设置IWDG重装载值

IWDG\_ReloadCounter(); //按照IWDG重装载寄存器的值重装载IWDG计数器

IWDG\_Enable(); //使能IWDG

}

//喂独立看门狗

void IWDG\_Feed(void)

{

IWDG\_ReloadCounter();//reload

}

## BlueTooth.c

#include "BlueTooth.h"

#include <string.h>

#include "usart.h"

#include "SimplePrintf.h"

#include "stdio.h"

#include "stdlib.h"

#include "Blue\_time.h"

#include "BlueData\_Deal.h"

#include "delay.h"

#include "communication\_to\_app.h"

#include "key.h"

#include "wakeup.h"

#include "oled.h"

u16 Blue\_time;

u8 Blue\_SendData\_Flag;

u8 Blue\_ReceveData\_Flag;

u8 Blue\_AT\_num;

u8 Blue\_ReceveData\_fail\_Flag;

char CONNECT\_MAC[12];

uint8\_t CONNECT\_NAME[14];

uint8\_t CONNECT\_NAME\_uint[7];

char Tuff[100];

uint8\_t BLE\_TO\_NB\_Data[100];

BlueConnect BlueConnectStruct;

void BLUE\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA|RCC\_APB2Periph\_GPIOC, ENABLE); // led4

GPIO\_InitStructure.GPIO\_Pin =GPIO\_Pin\_1; //

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

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_2MHz; //

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

GPIO\_ResetBits(GPIOA,GPIO\_Pin\_1);//蓝牙的电源控制引脚

GPIO\_InitStructure.GPIO\_Pin =GPIO\_Pin\_5|GPIO\_Pin\_7; //

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

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_2MHz; //

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

GPIO\_ResetBits(GPIOA,GPIO\_Pin\_1|GPIO\_Pin\_3); //PC1为蓝牙的使能引脚\*\*\*PC3为蓝牙的复位引脚

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

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPD; //

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_2MHz; //

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

//蓝牙接收数据引脚/\*\*\*\*接收到数据时为高电平，无数据时为低电平\*\*\*\*\*/

GPIO\_InitStructure.GPIO\_Pin =GPIO\_Pin\_6; //

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPD; //

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_2MHz; //

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

//连接指示灯\*\*\*\*\*/

}

void BLUE\_POWER\_OPEN(void)//开启蓝牙电源

{

BLUE\_POWER=1;

}

void BLUE\_POWER\_CLOSE(void)//关闭蓝牙电源

{

BLUE\_POWER=0;

}

void BLUE\_EN\_ENABLE(void)//蓝牙使能

{

BLUE\_EN=1;

}

void BLUE\_EN\_DISABLE(void)//蓝牙失能

{

BLUE\_EN=0;

}

void BLUE\_RST\_ENABLE(void)//蓝牙复位

{

BLUE\_RST=0;

}

void BLUE\_RST\_DISABLE(void)//蓝牙不复位

{

BLUE\_RST=1;

}

void Blue\_Flag\_Init(void)

{

Blue\_AT\_num =0;

Blue\_ReceveData\_Flag =1;

Blue\_SendData\_Flag =1;

Blue\_time =0;

}

void Blue\_S\_AT\_Config(void)//从模式的AT配置

{

if(Blue\_SendData\_Flag ==1)

{

Blue\_SendData\_Flag=0;

switch (Blue\_AT\_num)

{

case 0: OLED\_Clear();

OLED\_Display\_GB2312\_string(1,0,"正在启动蓝牙...");

BLUE\_EN\_ENABLE();

raw\_printf("当前蓝牙为从模式:");

Blue\_time =50;

break;

case 1:

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT+DEFAULT");//恢复出厂设置

Blue\_time =1;

break;

case 2: if(strstr(USART2\_RX\_Buff,"OK")!=NULL)

{

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT+ROLE=S");//设置为从模式

Blue\_time =10;

}

else

{

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT+DEFAULT");//恢复出厂设置

Blue\_AT\_num=1;

Blue\_time =10;

}

break;

case 3: if(strstr(USART2\_RX\_Buff,"OK")!=NULL)

{

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT+BAUD=115200");//设置波特率

Blue\_time =10;

}

break;

case 4: if(strstr(USART2\_RX\_Buff,"OK")!=NULL)

{

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT+NAME=NB\_BLUETOOTH—1");//设置蓝牙名字

Blue\_time =10;

}

break;

case 5: if(strstr(USART2\_RX\_Buff,"OK")!=NULL)

{

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT+RFPM=5");//设置蓝牙发送功率

Blue\_time =10;

}

break;

case 6: if(strstr(USART2\_RX\_Buff,"OK")!=NULL)

{

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT+MODE=0");//设置模式

Blue\_time =10;

}

break;

case 7: if(strstr(USART2\_RX\_Buff,"OK")!=NULL)

{

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT+RX");//查询蓝牙模块参数

Blue\_time =10;

BlueDataRceiveModeStruct.Blue\_AT\_Mode=0;

ConnectBlueStruct.ConnectBlueReceive\_AT=0;

BlueDataRceiveModeStruct.Blue\_To\_APP=1;

Work\_Struct.Mode\_type.Blue\_S.Con\_Time=240000L;

}

break;

case 8:

Blue\_Flag\_Init();

KEYStruct.KEY\_Time=0;

CommunicationStruct.S\_Mode\_Start =0;

KEYStruct.KEY\_S\_Mode\_Time=0;

LEDn\_ON(1);

OLED\_Clear();

OLED\_Display\_GB2312\_string(1,0,"蓝牙已启动！");

OLED\_Display\_GB2312\_string(1,2,"蓝牙名称：");

OLED\_Display\_GB2312\_string(1,4,"NB\_BLUETOOTH—1");

BlueConnectStruct.Connect=1;

break;

default: Blue\_time=0;

Blue\_AT\_num=0;

break;

}

}

}

//数字转字符串

void itoi(char \*str, unsigned long num)

{

char index[]="0123456789";

char i = 0, j = 0,temp;

do

{

str[i++] = index[num%10];

num /= 10;

}while(num);

str[i]='\0';

for(j=0; j<=(i-1)/2; j++)

{

temp=str[j];

str[j]=str[i-j-1];

str[i-j-1]=temp;

}

}

void BlueTooth\_S\_AT\_Send(void)

{

if(Blue\_SendData\_Flag&&Blue\_ReceveData\_Flag)

{

Blue\_ReceveData\_Flag=0;

Blue\_S\_AT\_Config();

}

}

void BLE\_StopScan\_Set(void)//停止扫描

{

// USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT+STOPSCAN");//

}

void BLE\_CLRBOND(void)

{

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT+CLRBOND");//取消绑定

}

u8 CheckDataSmae(char input[12],char compare[12])

{

u8 i=0,j=0,temp;

temp =12;

if(temp>1)

{

for(i=0;i<12;i++)

{

if(input[i] !=compare[i])

{

j=1;break;

}

}

}

return j;

}

## SimplePrintf.c

#include "BlueTooth.h"

#include <string.h>

#include "usart.h"

#include "SimplePrintf.h"

#include "stdio.h"

#include "stdlib.h"

#include "Blue\_time.h"

#include "BlueData\_Deal.h"

#include "delay.h"

#include "communication\_to\_app.h"

#include "key.h"

#include "wakeup.h"

#include "oled.h"

u16 Blue\_time;

u8 Blue\_SendData\_Flag;

u8 Blue\_ReceveData\_Flag;

u8 Blue\_AT\_num;

u8 Blue\_ReceveData\_fail\_Flag;

char CONNECT\_MAC[12];

uint8\_t CONNECT\_NAME[14];

uint8\_t CONNECT\_NAME\_uint[7];

char Tuff[100];

uint8\_t BLE\_TO\_NB\_Data[100];

BlueConnect BlueConnectStruct;

void BLUE\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA|RCC\_APB2Periph\_GPIOC, ENABLE); // led4

GPIO\_InitStructure.GPIO\_Pin =GPIO\_Pin\_1; //

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

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_2MHz; //

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

GPIO\_ResetBits(GPIOA,GPIO\_Pin\_1);//蓝牙的电源控制引脚

GPIO\_InitStructure.GPIO\_Pin =GPIO\_Pin\_5|GPIO\_Pin\_7; //

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

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_2MHz; //

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

GPIO\_ResetBits(GPIOA,GPIO\_Pin\_1|GPIO\_Pin\_3); //PC1为蓝牙的使能引脚\*\*\*PC3为蓝牙的复位引脚

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

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPD; //

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_2MHz; //

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

//蓝牙接收数据引脚/\*\*\*\*接收到数据时为高电平，无数据时为低电平\*\*\*\*\*/

GPIO\_InitStructure.GPIO\_Pin =GPIO\_Pin\_6; //

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPD; //

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_2MHz; //

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

//连接指示灯\*\*\*\*\*/

}

void BLUE\_POWER\_OPEN(void)//开启蓝牙电源

{

BLUE\_POWER=1;

}

void BLUE\_POWER\_CLOSE(void)//关闭蓝牙电源

{

BLUE\_POWER=0;

}

void BLUE\_EN\_ENABLE(void)//蓝牙使能

{

BLUE\_EN=1;

}

void BLUE\_EN\_DISABLE(void)//蓝牙失能

{

BLUE\_EN=0;

}

void BLUE\_RST\_ENABLE(void)//蓝牙复位

{

BLUE\_RST=0;

}

void BLUE\_RST\_DISABLE(void)//蓝牙不复位

{

BLUE\_RST=1;

}

void Blue\_Flag\_Init(void)

{

Blue\_AT\_num =0;

Blue\_ReceveData\_Flag =1;

Blue\_SendData\_Flag =1;

Blue\_time =0;

}

void Blue\_S\_AT\_Config(void)//从模式的AT配置

{

if(Blue\_SendData\_Flag ==1)

{

Blue\_SendData\_Flag=0;

switch (Blue\_AT\_num)

{

case 0: OLED\_Clear();

OLED\_Display\_GB2312\_string(1,0,"正在启动蓝牙...");

BLUE\_EN\_ENABLE();

raw\_printf("当前蓝牙为从模式:");

Blue\_time =50;

break;

case 1:

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT+DEFAULT");//恢复出厂设置

Blue\_time =1;

break;

case 2: if(strstr(USART2\_RX\_Buff,"OK")!=NULL)

{

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT+ROLE=S");//设置为从模式

Blue\_time =10;

}

else

{

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT+DEFAULT");//恢复出厂设置

Blue\_AT\_num=1;

Blue\_time =10;

}

break;

case 3: if(strstr(USART2\_RX\_Buff,"OK")!=NULL)

{

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT+BAUD=115200");//设置波特率

Blue\_time =10;

}

break;

case 4: if(strstr(USART2\_RX\_Buff,"OK")!=NULL)

{

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT+NAME=NB\_BLUETOOTH—1");//设置蓝牙名字

Blue\_time =10;

}

break;

case 5: if(strstr(USART2\_RX\_Buff,"OK")!=NULL)

{

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT+RFPM=5");//设置蓝牙发送功率

Blue\_time =10;

}

break;

case 6: if(strstr(USART2\_RX\_Buff,"OK")!=NULL)

{

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT+MODE=0");//设置模式

Blue\_time =10;

}

break;

case 7: if(strstr(USART2\_RX\_Buff,"OK")!=NULL)

{

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT+RX");//查询蓝牙模块参数

Blue\_time =10;

BlueDataRceiveModeStruct.Blue\_AT\_Mode=0;

ConnectBlueStruct.ConnectBlueReceive\_AT=0;

BlueDataRceiveModeStruct.Blue\_To\_APP=1;

Work\_Struct.Mode\_type.Blue\_S.Con\_Time=240000L;

}

break;

case 8:

Blue\_Flag\_Init();

KEYStruct.KEY\_Time=0;

CommunicationStruct.S\_Mode\_Start =0;

KEYStruct.KEY\_S\_Mode\_Time=0;

LEDn\_ON(1);

OLED\_Clear();

OLED\_Display\_GB2312\_string(1,0,"蓝牙已启动！");

OLED\_Display\_GB2312\_string(1,2,"蓝牙名称：");

OLED\_Display\_GB2312\_string(1,4,"NB\_BLUETOOTH—1");

BlueConnectStruct.Connect=1;

break;

default: Blue\_time=0;

Blue\_AT\_num=0;

break;

}

}

}

//数字转字符串

void itoi(char \*str, unsigned long num)

{

char index[]="0123456789";

char i = 0, j = 0,temp;

do

{

str[i++] = index[num%10];

num /= 10;

}while(num);

str[i]='\0';

for(j=0; j<=(i-1)/2; j++)

{

temp=str[j];

str[j]=str[i-j-1];

str[i-j-1]=temp;

}

}

void BlueTooth\_S\_AT\_Send(void)

{

if(Blue\_SendData\_Flag&&Blue\_ReceveData\_Flag)

{

Blue\_ReceveData\_Flag=0;

Blue\_S\_AT\_Config();

}

}

void BLE\_StopScan\_Set(void)//停止扫描

{

// USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT+STOPSCAN");//

}

void BLE\_CLRBOND(void)

{

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT+CLRBOND");//取消绑定

}

u8 CheckDataSmae(char input[12],char compare[12])

{

u8 i=0,j=0,temp;

temp =12;

if(temp>1)

{

for(i=0;i<12;i++)

{

if(input[i] !=compare[i])

{

j=1;break;

}

}

}

return j;

}

## Blue\_time.c

#include "Blue\_time.h"

#include "BlueTooth.h"

#include "usart.h"

#include "BlueData\_Deal.h"

#include "communication\_to\_app.h"

#include "SleepMode.h"

#include "wakeup.h"

//通用定时器4中断初始化

//这里时钟选择为APB1的2倍，而APB1为48M

//arr：自动重装值。

//psc：时钟预分频数

//这里使用的是定时器4!

void TIM4\_Init(u16 arr,u16 psc)

{

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;

NVIC\_InitTypeDef NVIC\_InitStructure;

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM4, ENABLE); //时钟使能

//定时器TIM4初始化

TIM\_TimeBaseStructure.TIM\_Period = arr; //设置在下一个更新事件装入活动的自动重装载寄存器周期的值

TIM\_TimeBaseStructure.TIM\_Prescaler =psc; //设置用来作为TIMx时钟频率除数的预分频值

TIM\_TimeBaseStructure.TIM\_ClockDivision = TIM\_CKD\_DIV1; //设置时钟分割:TDTS = Tck\_tim

TIM\_TimeBaseStructure.TIM\_CounterMode = TIM\_CounterMode\_Down; //TIM向下计数模式

TIM\_TimeBaseInit(TIM4, &TIM\_TimeBaseStructure); //根据指定的参数初始化TIMx的时间基数单位

TIM\_Cmd(TIM4, ENABLE); //

TIM\_ITConfig(TIM4,TIM\_IT\_Update,ENABLE ); //使能指定的TIM4中断,允许更新中断

//中断优先级NVIC设置

NVIC\_InitStructure.NVIC\_IRQChannel = TIM4\_IRQn; //TIM4中断

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 0; //先占优先级0级

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 2; //从优先级2级

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE; //IRQ通道被使能

NVIC\_Init(&NVIC\_InitStructure); //初始化NVIC寄存器

}

void OPEN\_TIME4(void)

{

TIM\_Cmd(TIM4, ENABLE); //使能TIM4

}

void CLOSE\_TIME4(void)

{

TIM\_Cmd(TIM4, DISABLE); //失能TIM4

}

//定时器4中断服务程序

void TIM4\_IRQHandler(void) //TIM4中断

{

if (TIM\_GetITStatus(TIM4, TIM\_IT\_Update) != RESET) //检查TIM4更新中断发生与否

{

TIM\_ClearITPendingBit(TIM4, TIM\_IT\_Update); //清除TIMx更新中断标志

if(ConnectBlueStruct.ReadData\_Time>0)

{

ConnectBlueStruct.ReadData\_Time--;

if(ConnectBlueStruct.ReadData\_Time==0)

{

ConnectBlueStruct.ReadBlueData\_fail=1;

}

}

if(ConnectBlueStruct.ScanTime>0)

{

ConnectBlueStruct.ScanTime--;

if(ConnectBlueStruct.ScanTime==0)

{

ConnectBlueStruct.Scan\_OverTime\_Flag=1;

}

}

if(Sleep\_Mode\_Struct.Enter\_SleepMode\_Time>0)

{

Sleep\_Mode\_Struct.Enter\_SleepMode\_Time--;

if(Sleep\_Mode\_Struct.Enter\_SleepMode\_Time==0)

{

Sleep\_Mode\_Struct.Enter\_SleepMode\_Flag=1;

}

}

if(CommunicationStruct.Receive\_Time>0)

{

CommunicationStruct.Receive\_Time--;

if(CommunicationStruct.Receive\_Time ==0)

{

CommunicationStruct.Receive\_Over\_Time=1;

}

}

if(Work\_Struct.Mode\_type.Blue\_S.Con\_Time>0)

{

Work\_Struct.Mode\_type.Blue\_S.Con\_Time--;

if(Work\_Struct.Mode\_type.Blue\_S.Con\_Time==0)

{

Work\_Struct.Mode\_type.Blue\_S.Con\_over\_Time=1;

}

}

}

}

## BlueData\_Deal.c

#include "BlueData\_Deal.h"

#include "usart.h"

#include "stdio.h"

#include "string.h"

#include "led.h"

#include "delay.h"

#include "BlueTooth.h"

#include "SimplePrintf.h"

#include "NB.h"

#include "flash.h"

#include "SleepMode.h"

#include "communication\_to\_app.h"

#include "wakeup.h"

uint8\_t yici=1;

u8 string\_12\_num=0;

u8 string\_14\_num=0;

u8 string\_14\_num\_2=0;

u8 ReceivMAC\_Flag;

u8 k=0;

u8 T=0;

char tempt1\_Read\_MAC[230];

char tempt2\_Read\_NAME[230];

char tempt3\_Read\_MAC[120];

char tempt4\_Read\_NAME[140];

char tempt5\_Read\_MAC[120];

char tempt6\_Read\_NAME[140];

char tempt7\_Read\_MAC[120];

char tempt8\_Read\_NAME[140];

char BlueNAME[255];//

char BlueMAC[216];//

char CurrentNAME\_String[14];

char CurrentMAC\_String[12];

char Connect\_Fail\_NAME\_Buff[255];

char Connect\_Fail\_MAC\_Buff[255];

char Fail\_NAME[14];

char Fail\_MAC[12];

uint8\_t ReadBuff[30];

uint8\_t StringConvert16\_BlueNAME[216];

ConnectBlueTypedef ConnectBlueStruct;

Connect\_Fail\_typedef Connect\_Fail\_Struct;

//ASCII转字符

void AscillToString(char \*InputData,char \*OutputData,u8 length)

{

u8 i;

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

{

if(InputData[i]>='0'&&InputData[i]<='9')

{

OutputData[i] = InputData[i] -'0';

}

else if(InputData[i]>='A'&&InputData[i]<='F')

{

OutputData[i] = InputData[i] -'A';

OutputData[i] += 10;

}

else if(InputData[i]>='a'&&InputData[i]<='f')

{

OutputData[i] = InputData[i] - 'a';

OutputData[i] += 10;

}

}

}

//16进制数组转字符串

void HEXArrayToStringArray(uint8\_t \*inputdata,char \*output,u16 length)

{

uint16\_t i =0;

uint8\_t TempBuff[254];

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

{

TempBuff[2\*i] = inputdata[i]>>4;

TempBuff[2\*i+1] = inputdata[i]&0x0f;

}

for(i=0;i<2\*length;i++)

{

sprintf(&output[i],"%X",TempBuff[i]);

}

}

//16进制字符串转数组

void StringArrayToHEXArray(char \*inputdata,char \*output,u16 length)

{

int i,n = 0;

for(i = 0; i<(length-1); i += 2)//数组长度含结束符1Byte故减去一个长度

{

if(inputdata[i] >= 'A' && inputdata[i] <= 'F')

output[n] = inputdata[i] - 'A' + 10;

else if(inputdata[i] >= 'a' && inputdata[i] <= 'f')

output[n] = inputdata[i] - 'a' + 10;

else

output[n] = inputdata[i] - '0';

if(inputdata[i + 1] >= 'A' && inputdata[i + 1] <= 'F')

output[n] = (output[n] << 4) | (inputdata[i + 1] - 'A' + 10);

else if(inputdata[i + 1] >= 'a' && inputdata[i + 1] <= 'f')

output[n] = (output[n] << 4) | (inputdata[i + 1] - 'a' + 10);

else

output[n] = (output[n] << 4) | (inputdata[i + 1] - '0');

++n;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*字符转16进制\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint16\_t StringToBytes( char\* pSrc, uint8\_t\* pDst, uint16\_t nSrcLength)

{

uint16\_t i = 0;

for (i = 0; i < nSrcLength; i += 2)

{

if ((\*pSrc >= '0') && (\*pSrc <= '9'))

{

\*pDst = (\*pSrc - '0') << 4;

}

else if((\*pSrc >= 'A') && (\*pSrc <= 'F'))

{

\*pDst = (\*pSrc - 'A' + 10) << 4;

}

else

{

\*pDst = (\*pSrc - 'a' + 10) << 4;

}

pSrc++;

if ((\*pSrc>='0') && (\*pSrc<='9'))

{

\*pDst |= (\*pSrc - '0');

}

else if((\*pSrc >= 'A') && (\*pSrc <= 'F'))

{

\*pDst |= (\*pSrc - 'A' + 10);

}

else

{

\*pDst |= (\*pSrc - 'a' + 10);

}

pSrc++;

pDst++;

}

return (nSrcLength / 2);

}

//十进制转BCD,length为BCD长度

void DectoBCD(int Dec,char \*Bcd, int length)

{

int i;

int temp;

for (i = length - 1; i >= 0; i--)

{

temp = Dec % 100;

Bcd[i] = ((temp / 10) << 4) + ((temp % 10) & 0x0F);

Dec /= 100;

}

}

// 查找字符串中，指定字符的位置

//F\_data为字符数据

//num为第几个。返回结果为0则表示无此字符。

uint8\_t find\_data(char \*dat , char F\_data , uint8\_t num )

{

char num\_data=0;

uint8\_t i=0;

for(i=0;i<255;i++)

{

if(dat[i] == F\_data)

{

num\_data++;

if(num\_data == num)

return i+1;

}

}

return 0;

}

//查找字符串被查找字符位置，返回值ff为没有，其他值为存在地址

u8 FindStringToString(char \*src, char \*sub)

{

u8 ls,la,i;

ls = strlen(src);//被查找值

la = strlen(sub);//查找值

if (la > ls)

return 0xff;

for (i=0;i<=ls-la;i++)

{

if(strncmp(src+i,sub,la)==0)

return i;

}

return 0xff;

}

//查找指定位置字符串中是否包含有该字符串

char FindStringToStringPlace(char \*src, char \*sub,u8 StartBit)

{

u8 i,j,k;

char TempBuff[255];

i=strlen(src);

for(j=0;j<i-StartBit;j++)

{

TempBuff[j] = src[StartBit+j];

}

k= FindStringToString(TempBuff, sub)+StartBit;

return k;

}

//查找数组中指定起始位指定长度是否含有某个元素

//u8 FindSameData(char \*input,char\*output,u8 startbit,u8 length)

//{

// char temp[length];

// u8 i,j;

// for(i=0;i<length;i++)

// { temp[i] = input[startbit+i]; }

// j= (strstr(temp, output) != NULL);

// return j;

//}

//主模式下蓝牙扫描的数据核对

void Check\_MAC(void)

{

if(USART2\_RX\_Buff[9]=='M'&&USART2\_RX\_Buff[10]=='A'&&USART2\_RX\_Buff[11]=='C')

{

if(USART2\_RX\_Buff[35]=='N'&&USART2\_RX\_Buff[36]=='A'&&USART2\_RX\_Buff[37]=='M'&&USART2\_RX\_Buff[38]=='E')

{

if(USART2\_RX\_Buff[26]=='R'&&USART2\_RX\_Buff[27]=='S')

{

if(USART2\_RX\_Buff[28]=='S'&&USART2\_RX\_Buff[29]=='I')

{

if(USART2\_RX\_Buff[40]=='J'&&USART2\_RX\_Buff[41]=='X')

{

ReceivMAC\_Flag=1;

}

}

}

}

}

}

void BLUE\_AT\_M\_Init(u8 Power)

{

char cmd[16],str[6];

itoi(str,Power);

strcpy(cmd, "AT+SCAN:RSSI=-");

strcat(cmd,str);

if(Blue\_SendData\_Flag ==1)

{

Blue\_SendData\_Flag=0;

switch (Blue\_AT\_num)

{

case 0: BLUE\_POWER\_OPEN();

BLUE\_EN\_ENABLE();

raw\_printf("启动扫描:");

Blue\_time =100;

break;

case 1:

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand(cmd);

break;

default:

break;

}

}

}

void SaveMAC\_NAME(uint8\_t MAC\_NUM,uint8\_t NAME\_NUM,uint8\_t NUM,u8 Power)

{

u8 j=0;

u8 i=0;

char cmd[16],str[6];

itoi(str,Power);

strcpy(cmd, "AT+SCAN:RSSI=-");

strcat(cmd,str);

for(j=0;j<50;j++)

{

if(USART2\_RX\_Buff[j-2]=='J'&&USART2\_RX\_Buff[j-1]=='X')

{

for(i=0;i<14;i++){CurrentNAME\_String[i]=USART2\_RX\_Buff[j++];}

}

}

if(strstr(BlueNAME,CurrentNAME\_String) == NULL)

{

for(j=0;j<60;j++)

{

if(USART2\_RX\_Buff[j-2]=='J'&&USART2\_RX\_Buff[j-1]=='X')

{

for(i=0;i<14;i++){BlueNAME[i+NAME\_NUM]=USART2\_RX\_Buff[j++];}

}

}

for(j=0;j<60;j++)

{

if(USART2\_RX\_Buff[j-4]=='M'&&USART2\_RX\_Buff[j-3]=='A'&&USART2\_RX\_Buff[j-2]=='C'&&USART2\_RX\_Buff[j-1]=='=')

{

for(i=0;i<12;i++){BlueMAC[i+MAC\_NUM]=USART2\_RX\_Buff[j++];}

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand(cmd);

}

}

}

else if(strstr(BlueNAME, CurrentNAME\_String) != NULL)

{

LED2=!LED2;

USart\_CLR\_RecvBuf(2);

memset(CurrentNAME\_String,0,sizeof(CurrentNAME\_String));

Uart2\_SendAtCommand(cmd);

k=NUM;

}

}

void Take\_Out\_NAME\_MAC(u8 Power)

{

u8 j=0;

u8 i=0;

char cmd[16],str[6];

itoi(str,Power);

strcpy(cmd, "AT+SCAN:RSSI=-");

strcat(cmd,str);

// BlueDataRceiveModeStruct.Blue\_AT\_Mode=1;

// ConnectBlueStruct.ConnectBlueReceive\_AT=1;

// BlueDataRceiveModeStruct.Blue\_To\_APP=0;

// BlueDataRceiveModeStruct.Blue\_Data\_Mode=0;

if(StringTimeTempt)

{

StringTimeTempt=0;

Check\_MAC();

if(!ReceivMAC\_Flag)

{

memset(BlueNAME,0,sizeof(BlueNAME));

memset(BlueMAC,0,sizeof(BlueMAC));

memset(CurrentNAME\_String,0,sizeof(CurrentNAME\_String));

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand(cmd);

T=0;

k=0;

ReceivMAC\_Flag=1;

}

if(ReceivMAC\_Flag)

{

k++;

switch (k)

{

case 1: SaveMAC\_NAME(0,0,0,80);

break;

case 2: SaveMAC\_NAME(12,14,1,80);

break;

case 3: SaveMAC\_NAME(24,28,2,80);

break;

case 4: SaveMAC\_NAME(36,42,3,80);

break;

case 5: SaveMAC\_NAME(48,56,4,80);

break;

case 6: SaveMAC\_NAME(60,70,5,80);

break;

case 7: SaveMAC\_NAME(72,84,6,80);

break;

case 8: SaveMAC\_NAME(84,98,7,80);

break;

case 9: SaveMAC\_NAME(96,112,8,80);

break;

case 10: SaveMAC\_NAME(108,126,9,80);

break;

case 11: SaveMAC\_NAME(120,140,10,80);

break;

case 12: SaveMAC\_NAME(132,154,11,80);

break;

case 13: SaveMAC\_NAME(144,168,12,80);

break;

case 14: SaveMAC\_NAME(156,182,13,80);

break;

case 15: SaveMAC\_NAME(168,196,14,80);

break;

case 16: SaveMAC\_NAME(180,210,15,80);

break;

case 17: SaveMAC\_NAME(192,224,16,80);

break;

case 18: for(j=0;j<12;j++){CurrentMAC\_String[j]=USART2\_RX\_Buff[j+13];}

for(j=0;j<14;j++){CurrentNAME\_String[j]=USART2\_RX\_Buff[j+42];}

if(strstr(BlueNAME, CurrentNAME\_String) == NULL&&strstr(BlueMAC, CurrentMAC\_String) == NULL)

{

for(j=0;j<60;j++)

{

if(USART2\_RX\_Buff[j-2]=='J'&&USART2\_RX\_Buff[j-1]=='X')

{

LED2=1;

for(i=0;i<14;i++){BlueNAME[i+238]=USART2\_RX\_Buff[j++];}

ConnectBlueStruct.ConnectBlueReceive\_AT=0;

k=19;

//break;

}

if(USART2\_RX\_Buff[j-4]=='M'&&USART2\_RX\_Buff[j-3]=='A'&&USART2\_RX\_Buff[j-2]=='C'&&USART2\_RX\_Buff[j-1]=='=')

{

for(i=0;i<12;i++){BlueMAC[i+204]=USART2\_RX\_Buff[j++];}

}

}

delay\_ms(300);

Uart3\_SendLR();

raw\_printf("扫描到的蓝牙表号:");

Uart3\_SendLR();

USART3\_Send\_xbit(BlueNAME,252,14);//252

Uart3\_SendLR();

raw\_printf("扫描到的蓝牙MAC地址:");

Uart3\_SendLR();

USART3\_Send\_xbit(BlueMAC,216,12);//216

Uart3\_SendLR();

USart\_CLR\_RecvBuf(2);

// /\*\*\*\*\*\*\*\*\*扫描到的表号及MAC写入FLASH中\*\*\*\*\*\*\*\*\*/

STMFLASH\_Write(SAVE\_ADDR\_NAME\_1,(u16\*)BlueNAME,140);

STMFLASH\_Write(SAVE\_ADDR\_MAC\_1,(u16\*)BlueMAC,120);

for(i=0;i<112;i++){BlueNAME[i]=BlueNAME[i+140];}

for(i=0;i<96;i++){BlueMAC[i]=BlueNAME[i+120];}

STMFLASH\_Write(SAVE\_ADDR\_NAME\_2,(u16\*)BlueNAME,112);

STMFLASH\_Write(SAVE\_ADDR\_MAC\_2,(u16\*)BlueMAC,96);

// /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CommunicationStruct.Start\_Scan1=0;

ConnectBlueStruct.Connect\_Number=0;

ConnectBlueStruct.StopScan=1;

}

else if(strstr(BlueNAME, CurrentNAME\_String) != NULL)

{

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand(cmd);

k=17;

}

break;

default:

break;

}

}

}

}

void Read\_Data\_Send(uint8\_t a,uint8\_t amc)//连接表端并读取表数据

{

u8 i;

char CONNECT[23];

if(BlueDataRceiveModeStruct.NB\_UpdateBlueData\_Success)

{

BlueDataRceiveModeStruct.Blue\_AT\_Mode=1;

BlueDataRceiveModeStruct.Blue\_Data\_Mode=0;

USart\_CLR\_RecvBuf(2);

Uart3\_SendLR();

Uart2\_SendAtCommand("AT+DISCONNECT");

memset(BlueDataRceiveModeStruct.DataMode.BlueDataBuff,0,sizeof(BlueDataRceiveModeStruct.DataMode.BlueDataBuff));

BlueDataRceiveModeStruct.NB\_UpdateBlueData\_Success=0;

ConnectBlueStruct.OK\_flag=1;

}

else if(strstr(USART2\_RX\_Buff,"OK")!=NULL||strstr(USART2\_RX\_Buff,"ERROR")!=NULL)//||strstr(USART2\_RX\_Buff,"ERROR")!=NULL

{

memset(CONNECT,0,sizeof(CONNECT));

memset(Fail\_NAME,0,sizeof(Fail\_NAME));

strcpy(CONNECT,"AT+CONNECT=");

for(i=0;i<12;i++){CONNECT[i+11]=BlueMAC[i+amc];}

for(i=0;i<12;i++){Fail\_MAC[i]=BlueMAC[i+amc];}

for(i=0;i<14;i++){Fail\_NAME[i]=BlueNAME[i+string\_14\_num\_2];}

string\_14\_num\_2+=14;

USart\_CLR\_RecvBuf(2);

Uart3\_SendLR();

Uart2\_SendAtCommand(CONNECT);

ConnectBlueStruct.Connect\_Time=3000;

ConnectBlueStruct.Connect\_Number=a;

}

}

void Read\_Data\_Update(uint8\_t a,uint8\_t name)//上传表数据

{

u8 i;

char CONNECT[23];

if(strstr(USART2\_RX\_Buff, "AT+CONNECTED") != NULL)//连接成功

{

ConnectBlueStruct.Connect\_Time=0;

ConnectBlueStruct.Connect\_OverTime\_Flag=0;

BlueDataRceiveModeStruct.Blue\_AT\_Mode=0;

BlueDataRceiveModeStruct.Blue\_To\_APP=0;

BlueDataRceiveModeStruct.Blue\_Data\_Mode=1;

memset(ReadBuff,0,sizeof(ReadBuff));

Read\_Meter\_19Protocol(name);

USART2\_Send\_Arr(ReadBuff,19);

USart\_CLR\_RecvBuf(2);

ConnectBlueStruct.ReadData\_Time=10000;

}

else if(ConnectBlueStruct.Connect\_OverTime\_Flag )

{

ConnectBlueStruct.ReadData\_Time=0;

ConnectBlueStruct.Connect\_OverTime\_Flag=0;

ConnectBlueStruct.ReadBlueData\_fail=0;

if(strstr(USART2\_RX\_Buff, "AT+CONNECTED") == NULL)//连接失败

{

ConnectBlueStruct.Connect\_fail\_Num++;

raw\_printf("连接表端失败,连接下一个");

memset(CONNECT,0,sizeof(CONNECT));

for(i=0;i<12;i++){Connect\_Fail\_MAC\_Buff[i+string\_12\_num]=Fail\_MAC[i];}

for(i=0;i<14;i++){Connect\_Fail\_NAME\_Buff[i+string\_14\_num]=Fail\_NAME[i];}

string\_12\_num+=12;

string\_14\_num+=14;

//BlueDataRceiveModeStruct.NB\_UpdateBlueData\_Success=1;

ConnectBlueStruct.Connect\_Number=a;

Uart3\_SendLR();

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand("AT");

delay\_ms(200);

Uart2\_SendAtCommand("AT");

}

}

else if(ConnectBlueStruct.ReadBlueData\_fail)

{

ConnectBlueStruct.ReadData\_Time=0;

ConnectBlueStruct.ReadBlueData\_fail=0;

ConnectBlueStruct.Connect\_Time=0;

ConnectBlueStruct.Connect\_fail\_Num++;

raw\_printf("读数据失败,连接下一个");

memset(CONNECT,0,sizeof(CONNECT));

BlueDataRceiveModeStruct.NB\_UpdateBlueData\_Success=1;

ConnectBlueStruct.Connect\_Number=a;

USart\_CLR\_RecvBuf(2);

// Uart3\_SendLR();

// Uart2\_SendAtCommand("DISCONNECT");

}

else if(ConnectBlueStruct.ReadBlueData\_Success)

{

ConnectBlueStruct.ReadBlueData\_fail=0;

ConnectBlueStruct.ReadData\_Time=0;

ConnectBlueStruct.ReadBlueData\_Success=0;

USart\_CLR\_RecvBuf(1);

NB\_Update\_Data();

}

else if(strstr(USART1\_RX\_Buff,"OK")!=NULL)

{

BlueDataRceiveModeStruct.NB\_UpdateBlueData\_Success=1;

ConnectBlueStruct.Connect\_Number=a;

//Uart2\_SendAtCommand("DISCONNECT");

delay\_ms(200);

USart\_CLR\_RecvBuf(1);

}

else if(strstr(USART1\_RX\_Buff,"ERROR")!=NULL)

{

USart\_CLR\_RecvBuf(1);

BlueDataRceiveModeStruct.NB\_UpdateBlueData\_Success=1;

ConnectBlueStruct.Connect\_Number=a;

}

}

void Connect\_Fail\_Deal(void)

{

u8 i;

if(Connect\_Fail\_Struct.Fail\_Num==1&&yici==1)

{

memset(BlueNAME,0,sizeof(BlueNAME));

memset(BlueMAC,0,sizeof(BlueMAC));

for(i=0;i<strlen(Connect\_Fail\_NAME\_Buff);i++){BlueNAME[i]=Connect\_Fail\_NAME\_Buff[i];}

for(i=0;i<strlen(Connect\_Fail\_MAC\_Buff);i++){BlueMAC[i]=Connect\_Fail\_MAC\_Buff[i];}

StringToBytes(BlueNAME,StringConvert16\_BlueNAME,strlen(Connect\_Fail\_NAME\_Buff));//蓝牙名称字符串转16进制

memset(Connect\_Fail\_NAME\_Buff,0,sizeof(Connect\_Fail\_NAME\_Buff));

memset(Connect\_Fail\_MAC\_Buff,0,sizeof(Connect\_Fail\_MAC\_Buff));

Connect\_Fail\_Struct.Fail\_Lenth=(strlen(Connect\_Fail\_NAME\_Buff))/14\*2;

ConnectBlueStruct.Connect\_quality=Connect\_Fail\_Struct.Fail\_Lenth;

ConnectBlueStruct.connect\_flag=1;

ConnectBlueStruct.BlueScanSuccess=1;

ConnectBlueStruct.Connect\_Number=0;

ConnectBlueStruct.Connect\_fail\_Num=0;

string\_14\_num\_2=0;

string\_12\_num=0;

string\_14\_num=0;

Connect\_Fail\_Struct.Fail\_Num=0;

ConnectBlueStruct.Connect\_fail\_Num=0;

yici=2;

}

else if(yici==2)

{

if(ConnectBlueStruct.Connect\_quality==ConnectBlueStruct.Connect\_Number)

{

ConnectBlueStruct.connect\_flag=0;

Work\_Struct.Mode\_type.JQZ\_Mode.Update++;

ConnectBlueStruct.Connect\_Number=0;

yici=0;

raw\_printf("读取表端数据结束");

Uart2\_SendAtCommand("AT+DISCONNECT");

delay\_ms(100);

if(BLUE\_CONNECT\_COND==1)

{

Uart2\_SendAtCommand("AT+DISCONNECT");

}

Uart2\_SendAtCommand("AT");

}

}

}

void Connect\_Blue\_Read\_Data(u8 jiedianshu)//读表数据

{

u8 i;

char CONNECT[23];

if(ConnectBlueStruct.Connect\_quality==ConnectBlueStruct.Connect\_Number&&yici==1)

{

ConnectBlueStruct.Connect\_Number=100;

memset(Fail\_NAME,0,sizeof(Fail\_NAME));

// Uart3\_SendLR();

// raw\_printf("读取表数据失败的表号和MAC地址:");

// USART3\_Send\_xbit(Connect\_Fail\_NAME\_Buff,strlen(Connect\_Fail\_NAME\_Buff),14);

// Uart3\_SendLR();

// USART3\_Send\_xbit(Connect\_Fail\_MAC\_Buff,strlen(Connect\_Fail\_MAC\_Buff),12);

Connect\_Fail\_Struct.Fail\_Num=1;

Sleep\_Mode\_Struct.Enter\_SleepMode\_Time=10000;

}

switch(ConnectBlueStruct.Connect\_Number)

{

case 0:

if(ConnectBlueStruct.BlueScanSuccess)//ConnectBlueStruct.BlueScanSuccess

{

memset(BlueDataRceiveModeStruct.DataMode.BlueDataStringBuff,0,sizeof(BlueDataRceiveModeStruct.DataMode.BlueDataStringBuff));

BlueDataRceiveModeStruct.Blue\_AT\_Mode=1;

BlueDataRceiveModeStruct.Blue\_Data\_Mode=0;

USart\_CLR\_RecvBuf(2);

Uart3\_SendLR();

raw\_printf("开始连接表端");

Uart2\_SendAtCommand("AT+ROLE=M");//CLRBOND

ConnectBlueStruct.BlueScanSuccess=0;

}

else if(strstr(USART2\_RX\_Buff,"OK")!=NULL)

{

memset(CONNECT,0,sizeof(CONNECT));

memset(Fail\_NAME,0,sizeof(Fail\_NAME));

strcpy(CONNECT,"AT+CONNECT=");

for(i=0;i<12;i++){CONNECT[i+11]=BlueMAC[i];}

for(i=0;i<12;i++){Fail\_MAC[i]=BlueMAC[i];}

for(i=0;i<14;i++){Fail\_NAME[i]=BlueNAME[i+string\_14\_num\_2];}

string\_14\_num\_2+=14;

Uart3\_SendLR();

USart\_CLR\_RecvBuf(2);

Uart2\_SendAtCommand(CONNECT);

Uart3\_SendLR();

memset(CONNECT,0,sizeof(CONNECT));

ConnectBlueStruct.Connect\_Time=4000;

ConnectBlueStruct.Connect\_Number=1;

memset(BlueDataRceiveModeStruct.DataMode.BlueDataBuff,0,sizeof(BlueDataRceiveModeStruct.DataMode.BlueDataBuff));

}

break;

case 1: Read\_Data\_Update(2,6);

break;

case 2: Read\_Data\_Send(3,12);

break;

case 3: Read\_Data\_Update(4,13);

break;

case 4: Read\_Data\_Send(5,24);

break;

case 5: Read\_Data\_Update(6,20);

break;

case 6: Read\_Data\_Send(7,36);

break;

case 7: Read\_Data\_Update(8,27);

break;

case 8: Read\_Data\_Send(9,48);

break;

case 9: Read\_Data\_Update(10,34);

break;

case 10: Read\_Data\_Send(11,60);

break;

case 11: Read\_Data\_Update(12,41);

break;

case 12: Read\_Data\_Send(13,72);

break;

case 13: Read\_Data\_Update(14,48);

break;

case 14: Read\_Data\_Send(15,84);

break;

case 15: Read\_Data\_Update(16,55);

break;

case 16: Read\_Data\_Send(17,96);

break;

case 17: Read\_Data\_Update(18,62);

break;

case 18: Read\_Data\_Send(19,108);

break;

// case 19: Read\_Data\_Update(20,69);

// break;

// case 20: Read\_Data\_Send(21,120);

// break;

//// case 21: Read\_Data\_Update(22,76);

// break;

// case 22: Read\_Data\_Send(23,132);

// break;

// case 23: Read\_Data\_Update(24,83);

// break;

// case 24: Read\_Data\_Send(25,144);

// break;

// case 25: Read\_Data\_Update(26,90);

// break;

// case 26: Read\_Data\_Send(27,156);

// break;

// case 27: Read\_Data\_Update(28,97);

// break;

// case 28: Read\_Data\_Send(29,168);

// break;

// case 29: Read\_Data\_Update(30,104);

// break;

// case 30: Read\_Data\_Send(31,180);

// break;

// case 31: Read\_Data\_Update(32,111);

// break;

// case 32: Read\_Data\_Send(33,192);

// break;

// case 33: Read\_Data\_Update(34,118);

// break;

// case 34: Read\_Data\_Send(35,204);

// break;

case 19: //Read\_Data\_Update(36,125);

if(strstr(USART2\_RX\_Buff, "AT+CONNECTED") != NULL)//连接成功

{

ConnectBlueStruct.Connect\_Time=0;

BlueDataRceiveModeStruct.Blue\_AT\_Mode=0;

BlueDataRceiveModeStruct.Blue\_Data\_Mode=1;

//memset(ReadBuff,0,sizeof(ReadBuff));

Read\_Meter\_19Protocol(69);

USART2\_Send\_Arr(ReadBuff,19);

USart\_CLR\_RecvBuf(2);

ConnectBlueStruct.ReadData\_Time=10000;

}

else if(ConnectBlueStruct.Connect\_OverTime\_Flag )

{

ConnectBlueStruct.Connect\_OverTime\_Flag=0;

if(strstr(USART2\_RX\_Buff, "AT+CONNECTED") == NULL)//连接失败

{

//Work\_Struct.Mode\_type.JQZ\_Mode.Update++;

//ConnectBlueStruct.Connect\_fail\_Num++;

Uart3\_SendLR();

raw\_printf("连接蓝牙失败");

printf("%d",ConnectBlueStruct.Connect\_fail\_Num);

printf("个");

memset(Fail\_NAME,0,sizeof(Fail\_NAME));

for(i=0;i<12;i++){Connect\_Fail\_MAC\_Buff[i+string\_12\_num]=Fail\_MAC[i];}

string\_14\_num+=14;

for(i=0;i<14;i++){Connect\_Fail\_NAME\_Buff[i+string\_14\_num]=Fail\_NAME[i];}

raw\_printf("读取表数据失败的表号和MAC地址:\r\n");

USART3\_Send\_xbit(Connect\_Fail\_NAME\_Buff,strlen(Connect\_Fail\_NAME\_Buff),14);

Uart3\_SendLR();

USART3\_Send\_xbit(Connect\_Fail\_MAC\_Buff,strlen(Connect\_Fail\_MAC\_Buff),12);

memset(CONNECT,0,sizeof(CONNECT));

BlueDataRceiveModeStruct.NB\_UpdateBlueData\_Success=1;

ConnectBlueStruct.Connect\_Number=20;

Connect\_Fail\_Struct.Fail\_Num=1;

yici=1;

}

}

else if(ConnectBlueStruct.ReadBlueData\_fail)

{

USART3\_Send\_xbit(Connect\_Fail\_MAC\_Buff,strlen(Connect\_Fail\_MAC\_Buff),12);

ConnectBlueStruct.ReadBlueData\_fail=0;

ConnectBlueStruct.Connect\_fail\_Num++;

memset(CONNECT,0,sizeof(CONNECT));

BlueDataRceiveModeStruct.NB\_UpdateBlueData\_Success=1;

ConnectBlueStruct.Connect\_Number=20;

USart\_CLR\_RecvBuf(2);

}

else if(ConnectBlueStruct.ReadBlueData\_Success)

{

ConnectBlueStruct.ReadData\_Time=0;

USart\_CLR\_RecvBuf(1);

NB\_Update\_Data();

ConnectBlueStruct.ReadBlueData\_Success=0;

}

else if(strstr(USART1\_RX\_Buff,"OK")!=NULL)

{

Uart3\_SendLR();

raw\_printf("读取表数据失败的表号和MAC地址:\r\n");

USART3\_Send\_xbit(Connect\_Fail\_NAME\_Buff,strlen(Connect\_Fail\_NAME\_Buff),14);

USART3\_Send\_xbit(Connect\_Fail\_MAC\_Buff,strlen(Connect\_Fail\_MAC\_Buff),12);

Uart2\_SendAtCommand("AT+DISCONNECT");

Connect\_Fail\_Struct.Fail\_Num=1;

yici=1;

delay\_ms(1000);

if(BLUE\_CONNECT\_COND==1)

{

Uart2\_SendAtCommand("AT+DISCONNECT");

}

//Work\_Struct.Mode\_type.JQZ\_Mode.Update++;

ConnectBlueStruct.Connect\_Number=20;

}

break;

default: //ConnectBlueStruct.Connect\_Number=100;

break;

}

}

extern char NAME[252];

extern char MAC[216];

void Read\_NAME\_AND\_NAME(void)//从FLASH中读取表号和MAC地址

{

uint8\_t i;

STMFLASH\_Read(SAVE\_ADDR\_NAME\_1,(u16\*)tempt2\_Read\_NAME,70);

STMFLASH\_Read(SAVE\_ADDR\_MAC\_1,(u16\*)tempt1\_Read\_MAC,60);

STMFLASH\_Read(SAVE\_ADDR\_MAC\_2,(u16\*)tempt3\_Read\_MAC,60);

STMFLASH\_Read(SAVE\_ADDR\_NAME\_2,(u16\*)tempt4\_Read\_NAME,70);

STMFLASH\_Read(SAVE\_ADDR\_MAC\_3,(u16\*)tempt5\_Read\_MAC,60);

STMFLASH\_Read(SAVE\_ADDR\_NAME\_3,(u16\*)tempt6\_Read\_NAME,70);

STMFLASH\_Read(SAVE\_ADDR\_MAC\_4,(u16\*)tempt7\_Read\_MAC,60);

STMFLASH\_Read(SAVE\_ADDR\_NAME\_4,(u16\*)tempt8\_Read\_NAME,70);

raw\_printf("读到的表号和MAC地址:");

raw\_printf("\r\n");

raw\_printf("第1-10个：");

USART3\_Send\_xbit(tempt2\_Read\_NAME,140,14);//252

USART3\_Send\_xbit(tempt1\_Read\_MAC,120,12);//252

raw\_printf("\r\n");

raw\_printf("第11-20个：");

USART3\_Send\_xbit(tempt4\_Read\_NAME,140,14);//252

USART3\_Send\_xbit(tempt3\_Read\_MAC,120,12);//252

raw\_printf("\r\n");

raw\_printf("第21-30个：");

USART3\_Send\_xbit(tempt6\_Read\_NAME,140,14);//252

USART3\_Send\_xbit(tempt5\_Read\_MAC,120,12);//252

raw\_printf("\r\n");

raw\_printf("第31-40个：");

USART3\_Send\_xbit(tempt6\_Read\_NAME,140,14);//252

USART3\_Send\_xbit(tempt5\_Read\_MAC,120,12);//252

for(i=0;i<140;i++){BlueNAME[i]=tempt2\_Read\_NAME[i];}

for(i=0;i<120;i++){BlueMAC[i]=tempt1\_Read\_MAC[i];}

StringToBytes(BlueNAME,StringConvert16\_BlueNAME,140);//蓝牙名称字符串转16进制

ConnectBlueStruct.Read\_Flag=1;

}

//CS校验，输入数组名，数组长度，数据开始校验的地方

u8 CS\_Check(uint8\_t DataBuff[100],u8 length,u8 StartBit)//char DataBuff[100]

{

u8 i,j=0;

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

{

j+=DataBuff[i+StartBit];

}

\_\_nop();

return j;

}

//读表数据协议格式

void Read\_Meter\_19Protocol(uint8\_t QQ)

{

u8 i;

memset(ReadBuff,0,sizeof(ReadBuff));

ReadBuff[0] = 0xfe;

ReadBuff[1] = 0xfe;

ReadBuff[2] = 0xfe;

ReadBuff[3] = 0x68;//包头

ReadBuff[4] = 0x10;//表类型

for(i=0;i<7;i++)

{ReadBuff[5+i] = StringConvert16\_BlueNAME[QQ-i];}

ReadBuff[12] = 0x01;//控制字

ReadBuff[13] = 0x03;//数据长度

ReadBuff[14] = 0x90;

ReadBuff[15] = 0x2f;

ReadBuff[16] = 0x00;//SER

ReadBuff[17] = CS\_Check(ReadBuff,14,3);

ReadBuff[18] = 0x16;

}

## Communication\_to\_app.c

#include "communication\_to\_app.h"

#include "key.h"

#include "BlueTooth.h"

#include "BlueData\_Deal.h"

#include "NB.h"

#include "delay.h"

#include "SimplePrintf.h"

#include "usart.h"

#include "flash.h"

#include "string.h"

#include "stdio.h"

#include "wakeup.h"

void KEY\_Set\_Blue\_S\_Mode(void)//按键设置蓝牙模式为从模式

{

if(KEY\_Scan(1)==2)

{

Work\_Struct.Mode\_type.Weakup.WeakUP\_Time=100;

Work\_Struct.Mode\_type.Sleep\_Mode=0;

KEYStruct.KEY\_Time++;

Work\_Struct.Mode\_type.Weakup.WeakUP\_Time=0;

Work\_Struct.Mode\_type.Weakup.WeakUP\_Over\_Time =0;//&&NB\_Struct.NB\_Init==1

if(KEYStruct.KEY\_Time==50)

{

weak\_BLUE\_init();

Sleep\_Mode\_Struct.WeakUP=0;

BlueDataRceiveModeStruct.Blue\_AT\_Mode=1;

ConnectBlueStruct.ConnectBlueReceive\_AT=0;

BLUE\_POWER\_OPEN();

BLUE\_EN\_ENABLE();

CommunicationStruct.S\_Mode\_Start=1;

KEYStruct.KEY\_Time=0;

}

}

if(CommunicationStruct.S\_Mode\_Start)

{

Work\_Struct.Mode\_type.Sleep\_Mode=0;

Blue\_S\_AT\_Config();

}

}

void Read\_Device\_information(void)//返回设备信息

{

u8 i;

uint8\_t Tuff[30];

char information[18];

memset(Device\_information\_u8\_Buff,0,sizeof(Device\_information\_u8\_Buff));

STMFLASH\_Read(FLASH\_SAVE\_ADDR+4096,(u16\*)information,18);

StringToBytes(information,Device\_information\_u8\_Buff,18);

Tuff[0]=0x68;

Tuff[1]=0x20;

for(i=0;i<7;i++){Tuff[2+i]=0xaa;}

Tuff[9]=0x32;

Tuff[10]=0x0d;

Tuff[11]=0x90;

Tuff[12]=0x2f;

Tuff[13]=0x00;

Tuff[14]=Device\_information\_u8\_Buff[0];

Tuff[15]=Device\_information\_u8\_Buff[1];

Tuff[16]=Device\_information\_u8\_Buff[2];

Tuff[17]=Device\_information\_u8\_Buff[3];

Tuff[18]=Device\_information\_u8\_Buff[4];

Tuff[19]=Device\_information\_u8\_Buff[5];

Tuff[20]=Device\_information\_u8\_Buff[6];

Tuff[21]=Device\_information\_u8\_Buff[7];

//for(j=0;j<8;j++){Tuff[14+j]=Device\_information\_u8\_Buff[j];}

Tuff[22]=Device\_information\_u8\_Buff[8];

Tuff[23]=0x00;//SER

Tuff[24] = CS\_Check(Tuff,23,0);

Tuff[25]=0x16;

// USart\_CLR\_RecvBuf(3);

// USART3\_Send\_Arr(Tuff,25);

memset(CommunicationStruct.CommunicationBuff,0,sizeof(CommunicationStruct.CommunicationBuff));

USART2\_Send\_Arr(Tuff,26);

CommunicationStruct.Send\_Device\_information=0;

}

void Scan\_Answer(void)//集中器扫描回复

{

u8 i;

uint8\_t Tuff[30];

Tuff[0]=0x68;

Tuff[1]=0x20;

for(i=0;i<7;i++){Tuff[2+i]=0xaa;}

Tuff[9]=0x33;

Tuff[10]=0x05;

Tuff[11]=0x90;

Tuff[12]=0x3f;

Tuff[13]=0x00;

Tuff[14]=0xbb;

Tuff[15]=0x00;//SER

Tuff[16]=CS\_Check(Tuff,16,0);

Tuff[17]=0x16;

// USart\_CLR\_RecvBuf(3);

// USART3\_Send\_Arr(Tuff,18);

memset(CommunicationStruct.CommunicationBuff,0,sizeof(CommunicationStruct.CommunicationBuff));

USART2\_Send\_Arr(Tuff,18);

}

void Read\_name\_mac(u8 ser)//集中器返回表号和MAC

{

u8 i;

uint8\_t Tuff[200];

uint8\_t BlueNAME\_u8[72];

uint8\_t BlueMAC\_u8[62];

char name[142];

char mac[122];

memset(BlueNAME\_u8,0,sizeof(BlueNAME\_u8));

memset(BlueMAC\_u8,0,sizeof(BlueMAC\_u8));

memset(name,0,sizeof(name));

memset(mac,0,sizeof(mac));

switch(ser)

{

case 0x01:

{

STMFLASH\_Read(SAVE\_ADDR\_NAME\_1,(u16\*)name,142);

STMFLASH\_Read(SAVE\_ADDR\_MAC\_1,(u16\*)mac,122);

}

break;

case 0x02:

{

STMFLASH\_Read(SAVE\_ADDR\_NAME\_2,(u16\*)name,142);

STMFLASH\_Read(SAVE\_ADDR\_MAC\_2,(u16\*)mac,122);

}

break;

case 0x03:

{

STMFLASH\_Read(SAVE\_ADDR\_NAME\_3,(u16\*)name,142);

STMFLASH\_Read(SAVE\_ADDR\_MAC\_3,(u16\*)mac,122);

}

break;

case 0x04:

{

STMFLASH\_Read(SAVE\_ADDR\_NAME\_4,(u16\*)name,142);

STMFLASH\_Read(SAVE\_ADDR\_MAC\_4,(u16\*)mac,122);

}

case 0x05:

{

// for(i=0;i<142;i++){name[i]='0';}

// for(i=0;i<122;i++){mac[i]='0';}

memset(name,'0',sizeof(name));

memset(mac,'0',sizeof(mac));

}

break;

default:

break;

}

StringToBytes(name,BlueNAME\_u8,142);

StringToBytes(mac,BlueMAC\_u8,122);

Tuff[0]=0x68;

Tuff[1]=0x20;

for(i=0;i<7;i++){Tuff[2+i]=0xaa;}

Tuff[9]=0x35;

Tuff[10]=0x86;

Tuff[11]=0x90;

Tuff[12]=0x5f;

Tuff[13]=0x00;

for(i=0;i<70;i++){Tuff[14+i]=BlueNAME\_u8[i];}

for(i=0;i<60;i++){Tuff[84+i]=BlueMAC\_u8[i];}

Tuff[144]=ser;//SER

Tuff[145]=CS\_Check(Tuff,144,0);

Tuff[146]=0x16;

memset(CommunicationStruct.CommunicationBuff,0,sizeof(CommunicationStruct.CommunicationBuff));

USART2\_Send\_Arr(Tuff,147);

}

void write\_name\_mac(u8 ser)//写表号后集中器返回

{

u8 i;

uint8\_t Tuff[30];

Tuff[0]=0x68;

Tuff[1]=0x20;

for(i=0;i<7;i++){Tuff[2+i]=0xaa;}

Tuff[9]=0x34;

Tuff[10]=0x06;

Tuff[11]=0x90;

Tuff[12]=0x4f;

Tuff[13]=0x00;

Tuff[14]=0x80;

Tuff[15]=0x00;

Tuff[16]=ser;//SER

Tuff[17]=CS\_Check(Tuff,16,0);

Tuff[18]=0x16;

// USart\_CLR\_RecvBuf(3);

// USART3\_Send\_Arr(Tuff,18);

memset(CommunicationStruct.CommunicationBuff,0,sizeof(CommunicationStruct.CommunicationBuff));

USART2\_Send\_Arr(Tuff,19);

}

void Set\_JZQ\_IP\_Return(void)//设置集中器IP及端口返回

{

u8 i;

uint8\_t Tuff[30];

Tuff[0]=0x68;

Tuff[1]=0x20;

for(i=0;i<7;i++){Tuff[2+i]=0xaa;}

Tuff[9]=0x95;

Tuff[10]=0x03;

Tuff[11]=0xB0;

Tuff[12]=0x19;

Tuff[13]=0x00;//SER

Tuff[14]=CS\_Check(Tuff,14,0);

Tuff[15]=0x16;

// USart\_CLR\_RecvBuf(3);

// USART3\_Send\_Arr(Tuff,18);

memset(CommunicationStruct.CommunicationBuff,0,sizeof(CommunicationStruct.CommunicationBuff));

USART2\_Send\_Arr(Tuff,16);

}

void Read\_JZQ\_IP\_COM(void)//读取集中器IP及端口返回

{

u8 i;

uint8\_t Tuff[50];

STMFLASH\_Read(FLASH\_SAVE\_ADDR+2048,(u16\*)CommunicationStruct.IP\_COM\_u8,20);

HEXArrayToStringArray((u8\*)CommunicationStruct.IP\_COM\_u8,CommunicationStruct.IP\_COM\_char,20);

Tuff[0]=0x68;

Tuff[1]=0x20;

for(i=0;i<7;i++){Tuff[2+i]=0xaa;}

Tuff[9]=0x83;

Tuff[10]=0x1a;

Tuff[11]=0x91;

Tuff[12]=0x0b;

Tuff[13]=0x00;

Tuff[14]=0x2c;

Tuff[15]=CommunicationStruct.IP\_COM\_u8[0];

Tuff[16]=CommunicationStruct.IP\_COM\_u8[1];

Tuff[17]=CommunicationStruct.IP\_COM\_u8[2];

Tuff[18]=CommunicationStruct.IP\_COM\_u8[3];

Tuff[19]=0x2c;

for(i=0;i<15;i++){Tuff[20+i]=CommunicationStruct.IP\_COM\_u8[i+5];}

Tuff[35]=0x00;

Tuff[36]=0x00;//SER

Tuff[37]=CS\_Check(Tuff,35,0);

Tuff[38]=0x16;

memset(CommunicationStruct.CommunicationBuff,0,sizeof(CommunicationStruct.CommunicationBuff));

USART2\_Send\_Arr(Tuff,39);

}

void Clear\_NAME\_MAC(void)//清表节点返回

{

u8 i;

uint8\_t Tuff[50];

Tuff[0]=0x68;

Tuff[1]=0x20;

for(i=0;i<7;i++){Tuff[2+i]=0xaa;}

Tuff[9]=0xa6;

Tuff[10]=0x05;

Tuff[11]=0xaa;

Tuff[12]=0xbb;

Tuff[13]=0x00;

Tuff[14]=0x00;

Tuff[15]=0x00;//SER

Tuff[16]=CS\_Check(Tuff,16,0);

Tuff[17]=0x16;

memset(CommunicationStruct.CommunicationBuff,0,sizeof(CommunicationStruct.CommunicationBuff));

USART2\_Send\_Arr(Tuff,18);

//68 20 AA AA AA AA AA AA AA A6 05 AA BB 00 00 00 3E 16

}

void Error\_Return(void)

{

u8 i;

uint8\_t Tuff[50];

Tuff[0]=0x68;

Tuff[1]=0x20;

for(i=0;i<7;i++){Tuff[2+i]=0xaa;}

Tuff[9]=0x00;

Tuff[10]=0x05;

Tuff[11]=0xff;

Tuff[12]=0xff;

Tuff[13]=0xff;

Tuff[14]=0xff;

Tuff[15]=0xff;//SER

Tuff[16]=CS\_Check(Tuff,16,0);

Tuff[17]=0x16;

memset(CommunicationStruct.CommunicationBuff,0,sizeof(CommunicationStruct.CommunicationBuff));

USART2\_Send\_Arr(Tuff,18);

}

void Determine\_Deveice(void)

{

if(CommunicationStruct.Receive\_Over\_Time)

{

if(CommunicationStruct.CommunicationBuff[CommunicationStruct.CommunicationBuff[10]+12]!=0x16)

{

OLED\_Clear();

OLED\_Display\_GB2312\_string(1,6,"接收数据错误！！");

Error\_Return();

CommunicationStruct.Communication\_RX\_Cont=0;

memset(CommunicationStruct.CommunicationBuff,0,sizeof(CommunicationStruct.CommunicationBuff));

CommunicationStruct.Receive\_Time=0;

CommunicationStruct.Receive\_Over\_Time=0;

}

}

}

void APP\_To\_BlueScan(uint32\_t time)//APP启动扫描

{

u8 i;

if(CommunicationStruct.Start)

{

BlueDataRceiveModeStruct.Blue\_AT\_Mode=2;

ConnectBlueStruct.ConnectBlueReceive\_AT=1;

BlueDataRceiveModeStruct.Blue\_To\_APP=0;

BlueDataRceiveModeStruct.Blue\_Data\_Mode=0;

ConnectBlueStruct.connect\_flag=0;

CommunicationStruct.Start\_Scan1=1;

BLUE\_POWER\_CLOSE();

BLUE\_EN\_DISABLE();

delay\_ms(500);

BLUE\_POWER\_OPEN();

BLUE\_EN\_ENABLE();

ConnectBlueStruct.ScanTime=time;

Uart2\_SendAtCommand("AT+ROLE=M");

delay\_ms(200);

Uart2\_SendAtCommand("AT+SCAN:RSSI=-80");

CommunicationStruct.Start=0;

}

if(CommunicationStruct.Start\_Scan1)

{

Take\_Out\_NAME\_MAC(80);

}

if(ConnectBlueStruct.Scan\_OverTime\_Flag)

{

Uart2\_SendAtCommand("AT+STOPSCAN");

delay\_ms(100);

Uart3\_SendLR();

printf("扫描结束,扫描到的表端数量：");

printf("%d",strlen(BlueNAME)/14);

CommunicationStruct.Start\_Scan1=0;

for(i=0;i<(252-strlen(BlueNAME));i++){BlueNAME[strlen(BlueNAME)+i]='0';}

for(i=0;i<(216-strlen(BlueMAC));i++){BlueMAC[strlen(BlueMAC)+i]='0';}

STMFLASH\_Write(SAVE\_ADDR\_NAME\_1,(u16\*)BlueNAME,140);

STMFLASH\_Write(SAVE\_ADDR\_MAC\_1,(u16\*)BlueMAC,120);

for(i=0;i<112;i++){BlueNAME[i]=BlueNAME[i+140];}

for(i=0;i<96;i++){BlueMAC[i]=BlueMAC[i+120];}

STMFLASH\_Write(SAVE\_ADDR\_NAME\_2,(u16\*)BlueNAME,112);

STMFLASH\_Write(SAVE\_ADDR\_MAC\_2,(u16\*)BlueMAC,96);

Blue\_Flag\_Init();

ConnectBlueStruct.Scan\_OverTime\_Flag=0;

}

}

void write\_mac\_name(uint8\_t ser,uint32\_t name\_addr,uint32\_t mac\_addr)

{

u8 i;

uint8\_t tempt\_name[142];

uint8\_t tempt\_mac[122];

char name\_222[142];

char mac\_222[122];

memset(name\_222,0,sizeof(name\_222));

memset(mac\_222,0,sizeof(mac\_222));

memset(tempt\_name,0,sizeof(tempt\_name));

memset(tempt\_mac,0,sizeof(tempt\_mac));

for(i=0;i<70;i++){tempt\_name[i]=CommunicationStruct.CommunicationBuff[14+i];}

for(i=0;i<60;i++){tempt\_mac[i]=CommunicationStruct.CommunicationBuff[84+i];}

HEXArrayToStringArray(tempt\_name,name\_222,71);

HEXArrayToStringArray(tempt\_mac,mac\_222,61);

STMFLASH\_Write(name\_addr,(u16\*)name\_222,142);

STMFLASH\_Write(mac\_addr,(u16\*)mac\_222,122);

CommunicationStruct.Communication\_RX\_Cont=0;

memset(CommunicationStruct.CommunicationBuff,0,sizeof(CommunicationStruct.CommunicationBuff));

write\_name\_mac(ser);//写表号返回

CommunicationStruct.write\_ser=0x00;

}

void JZQ\_Write\_NAME\_MAC(void)

{

switch(CommunicationStruct.write\_ser)

{

case 0x01:write\_mac\_name(0x01,SAVE\_ADDR\_NAME\_1,SAVE\_ADDR\_MAC\_1);

break;

case 0x02:write\_mac\_name(0x02,SAVE\_ADDR\_NAME\_2,SAVE\_ADDR\_MAC\_2);

break;

case 0x03:write\_mac\_name(0x03,SAVE\_ADDR\_NAME\_3,SAVE\_ADDR\_MAC\_3);

break;

case 0x04:write\_mac\_name(0x04,SAVE\_ADDR\_NAME\_4,SAVE\_ADDR\_MAC\_4);

break;

default:

break;

}

}

void APP\_TO\_JZQ\_CB\_RETURN(void)

{

u8 i;

uint8\_t Tuff[50];

Tuff[0]=0x68;

Tuff[1]=0x20;

for(i=0;i<7;i++){Tuff[2+i]=0xaa;}

Tuff[9]=0x36;

Tuff[10]=0x04;

Tuff[11]=0xc1;

Tuff[12]=0xb1;

Tuff[13]=0x11;

Tuff[14]=0x00;

Tuff[15]=CS\_Check(Tuff,15,0);

Tuff[16]=0x16;

memset(CommunicationStruct.CommunicationBuff,0,sizeof(CommunicationStruct.CommunicationBuff));

USART2\_Send\_Arr(Tuff,17);

}

## Oled.c

#include "oled.h"

#include "stdlib.h"

#include "delay.h"

#include "beep.h"

//反显函数

OLEDTypedef OLEDStruct;

void OLED\_ColorTurn(u8 i)

{

if(i==0)

{

OLED\_WR\_Byte(0xA6,OLED\_CMD);//正常显示

}

if(i==1)

{

OLED\_WR\_Byte(0xA7,OLED\_CMD);//反色显示

}

}

//屏幕旋转180度

void OLED\_DisplayTurn(u8 i)

{

if(i==0)

{

OLED\_WR\_Byte(0xC8,OLED\_CMD);//正常显示

OLED\_WR\_Byte(0xA1,OLED\_CMD);

}

if(i==1)

{

OLED\_WR\_Byte(0xC0,OLED\_CMD);//反转显示

OLED\_WR\_Byte(0xA0,OLED\_CMD);

}

}

//向SSD1306写入一个字节。

//mode:数据/命令标志 0,表示命令;1,表示数据;

void OLED\_WR\_Byte(u32 dat,u8 mode)

{

u8 i;

if(mode)

{

OLED\_DC\_Set(); //写数据

}

else

{

OLED\_DC\_Clr(); //写命令

}

OLED\_CS\_Clr();

for(i=0;i<8;i++)

{

OLED\_SCLK\_Clr();//将时钟信号设置为低电平

if(dat&0x80)//将dat的8位从最高位依次写入

{

OLED\_SDIN\_Set();

}

else

{

OLED\_SDIN\_Clr();

}

OLED\_SCLK\_Set();//将时钟信号设置为高电平

dat<<=1;

}

OLED\_CS\_Set();

OLED\_DC\_Set();

}

//清屏函数

void OLED\_Clear(void)

{

u8 i,n;

for(i=0;i<8;i++)

{

OLED\_WR\_Byte(0xb0+i,OLED\_CMD);

OLED\_WR\_Byte(0x10,OLED\_CMD);

OLED\_WR\_Byte(0x02,OLED\_CMD);

for(n=0;n<128;n++)

{

OLED\_WR\_Byte(0x00,OLED\_DATA);//清除所有数据

}

}

}

//设置起始地址

void OLED\_address(u8 x,u8 y)

{

OLED\_WR\_Byte(0xb0+y,OLED\_CMD); //设置页地址

OLED\_WR\_Byte((((x+2)&0xf0)>>4)|0x10,OLED\_CMD); //设置列地址的高4位

OLED\_WR\_Byte(((x+2)&0x0f),OLED\_CMD); //设置列地址的低4位

}

//显示128x64点阵图像

void OLED\_Display\_128x64(u8 \*dp)

{

u8 i,j;

for(i=0;i<8;i++)

{

OLED\_address(0,i);

for(j=0;j<128;j++)

{

OLED\_WR\_Byte(\*dp,OLED\_DATA); //写数据到OLED,每写完一个8位的数据后列地址自动加1

dp++;

}

}

}

//显示16x16点阵图像、汉字、生僻字或16x16点阵的其他图标

void OLED\_Display\_16x16(u32 x,u32 y,u8 \*dp)

{

u8 i,j;

for(j=0;j<2;j++)

{

OLED\_address(x,y);

for(i=0;i<16;i++)

{

OLED\_WR\_Byte(\*dp,OLED\_DATA); //写数据到LCD,每写完一个8位的数据后列地址自动加1

dp++;

}

y++;

}

}

//显示8x16点阵图像、ASCII, 或8x16点阵的自造字符、其他图标

void OLED\_Display\_8x16(u8 x,u32 y,u8 \*dp)

{

u8 i,j;

for(j=0;j<2;j++)

{

OLED\_address(x,y);

for(i=0;i<8;i++)

{

OLED\_WR\_Byte(\*dp,OLED\_DATA); //写数据到LCD,每写完一个8位的数据后列地址自动加1

dp++;

}

y++;

}

}

//显示5\*7点阵图像、ASCII, 或5x7点阵的自造字符、其他图标

void OLED\_Display\_5x7(u8 x,u32 y,u8 \*dp)

{

u8 i;

OLED\_address(x,y);

for(i=0;i<6;i++)

{

OLED\_WR\_Byte(\*dp,OLED\_DATA);

dp++;

}

}

//送指令到晶联讯字库IC

void Send\_Command\_to\_ROM(u8 dat)

{

u8 i;

for(i=0;i<8;i++)

{

OLED\_SCLK\_Clr();

if(dat&0x80)

{

OLED\_SDIN\_Set();

}

else

{

OLED\_SDIN\_Clr();

}

dat<<=1;

OLED\_SCLK\_Set();

}

}

//从晶联讯字库IC中取汉字或字符数据（1个字节）

u8 Get\_data\_from\_ROM()

{

u8 i,read=0;

for(i=0;i<8;i++)

{

OLED\_SCLK\_Clr();

read<<=1;

if(GPIO\_ReadInputDataBit(GPIOB,GPIO\_Pin\_5))

{

read=read+1;

}

else

{

read=read;

}

OLED\_SCLK\_Set();

}

return read;

}

//从相关地址（addrHigh：地址高字节,addrMid：地址中字节,addrLow：地址低字节）中连续读出DataLen个字节的数据到 pbuff的地址

//连续读取

void OLED\_get\_data\_from\_ROM(u8 addrHigh,u8 addrMid,u8 addrLow,u8 \*pbuff,u8 DataLen)

{

u8 i;

OLED\_ROM\_CS\_Clr();

OLED\_CS\_Set();

Send\_Command\_to\_ROM(0x03);

Send\_Command\_to\_ROM(addrHigh);

Send\_Command\_to\_ROM(addrMid);

Send\_Command\_to\_ROM(addrLow);

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

{

\*(pbuff+i)=Get\_data\_from\_ROM();

}

OLED\_ROM\_CS\_Set();

}

u8long fontaddr=0;

void OLED\_Display\_GB2312\_string(u8 x,u8 y,u8 \*text)

{

u8 i=0;

u8 addrHigh,addrMid,addrLow;

u8 fontbuf[32];

while(text[i]>0x00)

{

if((text[i]>=0xb0)&&(text[i]<=0xf7)&&(text[i+1]>=0xa1))

{

//国标简体（GB2312）汉字在晶联讯字库IC中的地址由以下公式来计算：

//Address = ((MSB - 0xB0) \* 94 + (LSB - 0xA1)+ 846)\*32+ BaseAdd;BaseAdd=0

//由于担心8位单片机有乘法溢出问题，所以分三部取地址

fontaddr=(text[i]-0xb0)\*94;

fontaddr+=(text[i+1]-0xa1)+846;

fontaddr=(u8long)(fontaddr\*32);

addrHigh=(fontaddr&0xff0000)>>16; //地址的高8位,共24位

addrMid=(fontaddr&0xff00)>>8; //地址的中8位,共24位

addrLow=(fontaddr&0xff); //地址的低8位,共24位

OLED\_get\_data\_from\_ROM(addrHigh,addrMid,addrLow,fontbuf,32);

//取32个字节的数据，存到"fontbuf[32]"

OLED\_Display\_16x16(x,y,fontbuf);

//显示汉字到LCD上，y为页地址，x为列地址，fontbuf[]为数据

x+=16;

i+=2;

}

else if((text[i]>=0xa1)&&(text[i]<=0xa3)&&(text[i+1]>=0xa1))

{

fontaddr=(text[i]-0xa1)\*94;

fontaddr+=(text[i+1]-0xa1);

fontaddr=(u8long)(fontaddr\*32);

addrHigh=(fontaddr&0xff0000)>>16;

addrMid=(fontaddr&0xff00)>>8;

addrLow=(fontaddr&0xff);

OLED\_get\_data\_from\_ROM(addrHigh,addrMid,addrLow,fontbuf,32);

OLED\_Display\_16x16(x,y,fontbuf);

x+=16;

i+=2;

}

else if((text[i]>=0x20)&&(text[i]<=0x7e))

{

unsigned char fontbuf[16];

fontaddr=(text[i]-0x20);

fontaddr=(unsigned long)(fontaddr\*16);

fontaddr=(unsigned long)(fontaddr+0x3cf80);

addrHigh=(fontaddr&0xff0000)>>16;

addrMid=(fontaddr&0xff00)>>8;

addrLow=fontaddr&0xff;

OLED\_get\_data\_from\_ROM(addrHigh,addrMid,addrLow,fontbuf,16);

OLED\_Display\_8x16(x,y,fontbuf);

x+=8;

i+=1;

}

else

i++;

}

}

void OLED\_Display\_string\_5x7(u8 x,u8 y,u8 \*text)

{

u8 i=0;

u8 addrHigh,addrMid,addrLow;

while(text[i]>0x00)

{

if((text[i]>=0x20)&&(text[i]<=0x7e))

{

u8 fontbuf[8];

fontaddr=(text[i]-0x20);

fontaddr=(unsigned long)(fontaddr\*8);

fontaddr=(unsigned long)(fontaddr+0x3bfc0);

addrHigh=(fontaddr&0xff0000)>>16;

addrMid=(fontaddr&0xff00)>>8;

addrLow=fontaddr&0xff;

OLED\_get\_data\_from\_ROM(addrHigh,addrMid,addrLow,fontbuf,8);

OLED\_Display\_5x7(x,y,fontbuf);

x+=6;

i+=1;

}

else

i++;

}

}

//显示2个数字

//x,y :起点坐标

//num1：要显示的小数

//len :数字的位数

void OLED\_ShowNum(u8 x,u8 y,float num1,u8 len)

{

u8 i;

u32 t,num;

x=x+len\*8+8;//要显示的小数最低位的横坐标

num=num1\*100;//将小数左移两位并转化为整数

OLED\_Display\_GB2312\_string(x-24,y,".");//显示小数点

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

{

t=num%10;//取个位数的数值

num=num/10;//将整数右移一位

x-=8;

if(i==2){x-=8;}//当显示出来两个小数之后，空出小数点的位置

switch(t)

{

case 0 :OLED\_Display\_GB2312\_string(x,y,"0");break;

case 1 :OLED\_Display\_GB2312\_string(x,y,"1");break;

case 2 :OLED\_Display\_GB2312\_string(x,y,"2");break;

case 3 :OLED\_Display\_GB2312\_string(x,y,"3");break;

case 4 :OLED\_Display\_GB2312\_string(x,y,"4");break;

case 5 :OLED\_Display\_GB2312\_string(x,y,"5");break;

case 6 :OLED\_Display\_GB2312\_string(x,y,"6");break;

case 7 :OLED\_Display\_GB2312\_string(x,y,"7");break;

case 8 :OLED\_Display\_GB2312\_string(x,y,"8");break;

case 9 :OLED\_Display\_GB2312\_string(x,y,"9");break;

}

}

}

//OLED的初始化

void OLED\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB|RCC\_APB2Periph\_AFIO, ENABLE); //使能A端口时钟

GPIO\_PinRemapConfig(GPIO\_Remap\_SWJ\_JTAGDisable, ENABLE);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_9|GPIO\_Pin\_8|GPIO\_Pin\_7|GPIO\_Pin\_6|GPIO\_Pin\_5|GPIO\_Pin\_4;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP; //推挽输出

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;//速度50MHz

GPIO\_Init(GPIOB, &GPIO\_InitStructure); //初始化GPIOD3,6

GPIO\_SetBits(GPIOB,GPIO\_Pin\_9|GPIO\_Pin\_8|GPIO\_Pin\_7|GPIO\_Pin\_6|GPIO\_Pin\_5|GPIO\_Pin\_4);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB|RCC\_APB2Periph\_AFIO, ENABLE);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_5;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPD; //推挽输出

GPIO\_Init(GPIOB, &GPIO\_InitStructure); //初始化GPIOA4

OLED\_WR\_Byte(0xAE,OLED\_CMD);//--turn off oled panel

OLED\_WR\_Byte(0x00,OLED\_CMD);//---set low column address

OLED\_WR\_Byte(0x10,OLED\_CMD);//---set high column address

OLED\_WR\_Byte(0x40,OLED\_CMD);//--set start line address Set Mapping RAM Display Start Line (0x00~0x3F)

OLED\_WR\_Byte(0x81,OLED\_CMD);//--set contrast control register

OLED\_WR\_Byte(0xCF,OLED\_CMD);// Set SEG Output Current Brightness

OLED\_WR\_Byte(0xA1,OLED\_CMD);//--Set SEG/Column Mapping 0xa0左右反置 0xa1正常

OLED\_WR\_Byte(0xC8,OLED\_CMD);//Set COM/Row Scan Direction 0xc0上下反置 0xc8正常

OLED\_WR\_Byte(0xA6,OLED\_CMD);//--set normal display

OLED\_WR\_Byte(0xA8,OLED\_CMD);//--set multiplex ratio(1 to 64)

OLED\_WR\_Byte(0x3f,OLED\_CMD);//--1/64 duty

OLED\_WR\_Byte(0xD3,OLED\_CMD);//-set display offset Shift Mapping RAM Counter (0x00~0x3F)

OLED\_WR\_Byte(0x00,OLED\_CMD);//-not offset

OLED\_WR\_Byte(0xd5,OLED\_CMD);//--set display clock divide ratio/oscillator frequency

OLED\_WR\_Byte(0x80,OLED\_CMD);//--set divide ratio, Set Clock as 100 Frames/Sec

OLED\_WR\_Byte(0xD9,OLED\_CMD);//--set pre-charge period

OLED\_WR\_Byte(0xF1,OLED\_CMD);//Set Pre-Charge as 15 Clocks & Discharge as 1 Clock

OLED\_WR\_Byte(0xDA,OLED\_CMD);//--set com pins hardware configuration

OLED\_WR\_Byte(0x12,OLED\_CMD);

OLED\_WR\_Byte(0xDB,OLED\_CMD);//--set vcomh

OLED\_WR\_Byte(0x40,OLED\_CMD);//Set VCOM Deselect Level

OLED\_WR\_Byte(0x20,OLED\_CMD);//-Set Page Addressing Mode (0x00/0x01/0x02)

OLED\_WR\_Byte(0x02,OLED\_CMD);//

OLED\_WR\_Byte(0x8D,OLED\_CMD);//--set Charge Pump enable/disable

OLED\_WR\_Byte(0x14,OLED\_CMD);//--set(0x10) disable

OLED\_WR\_Byte(0xA4,OLED\_CMD);// Disable Entire Display On (0xa4/0xa5)

OLED\_WR\_Byte(0xA6,OLED\_CMD);// Disable Inverse Display On (0xa6/a7)

OLED\_WR\_Byte(0xAF,OLED\_CMD);

}

## Beep.c

#include "beep.h"

void BEEP\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

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

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

GPIO\_InitStructure.GPIO\_Pin =GPIO\_Pin\_3;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_2MHz;

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

GPIO\_ResetBits(GPIOB,GPIO\_Pin\_3);

}

void BEEP\_ON(uint8\_t times)

{

u8 i;

if(times>0)

{

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

{

BEEP=1;

delay\_ms(40);

BEEP=0;

delay\_ms(40);

}

}

else

{

BEEP=0;

}

}

## Flash.c

#include "flash.h"

FLASH\_TYPEDEF FLASH\_Struct;

//读取指定地址的半字(16位数据)

//faddr:读地址(此地址必须为2的倍数!!)

//返回值:对应数据.

u16 STMFLASH\_ReadHalfWord(u32 faddr)

{

return \*(vu16\*)faddr;

}

#if STM32\_FLASH\_WREN //如果使能了写

//不检查的写入

//WriteAddr:起始地址

//pBuffer:数据指针

//NumToWrite:半字(16位)数

void STMFLASH\_Write\_NoCheck(u32 WriteAddr,u16 \*pBuffer,u16 NumToWrite)

{

u16 i;

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

{

FLASH\_ProgramHalfWord(WriteAddr,pBuffer[i]);

WriteAddr+=2;//地址增加2.

}

}

//从指定地址开始写入指定长度的数据

//WriteAddr:起始地址(此地址必须为2的倍数!!)

//pBuffer:数据指针

//NumToWrite:半字(16位)数(就是要写入的16位数据的个数.)

#if STM32\_FLASH\_SIZE<256

#define STM\_SECTOR\_SIZE 1024 //字节

#else

#define STM\_SECTOR\_SIZE 2048

#endif

u16 STMFLASH\_BUF[STM\_SECTOR\_SIZE/2];//最多是2K字节

void STMFLASH\_Write(u32 WriteAddr,u16 \*pBuffer,u16 NumToWrite)

{

u32 secpos; //扇区地址

u16 secoff; //扇区内偏移地址(16位字计算)

u16 secremain; //扇区内剩余地址(16位字计算)

u16 i;

u32 offaddr; //去掉0X08000000后的地址

if(WriteAddr<STM32\_FLASH\_BASE||(WriteAddr>=(STM32\_FLASH\_BASE+1024\*STM32\_FLASH\_SIZE)))return;//非法地址

FLASH\_Unlock(); //解锁

offaddr=WriteAddr-STM32\_FLASH\_BASE; //实际偏移地址.

secpos=offaddr/STM\_SECTOR\_SIZE; //扇区地址 0~127 for STM32F103RBT6

secoff=(offaddr%STM\_SECTOR\_SIZE)/2; //在扇区内的偏移(2个字节为基本单位.)

secremain=STM\_SECTOR\_SIZE/2-secoff; //扇区剩余空间大小

if(NumToWrite<=secremain)secremain=NumToWrite;//不大于该扇区范围

while(1)

{

STMFLASH\_Read(secpos\*STM\_SECTOR\_SIZE+STM32\_FLASH\_BASE,STMFLASH\_BUF,STM\_SECTOR\_SIZE/2);//读出整个扇区的内容

for(i=0;i<secremain;i++)//校验数据

{

if(STMFLASH\_BUF[secoff+i]!=0XFFFF)break;//需要擦除

}

if(i<secremain)//需要擦除

{

FLASH\_ErasePage(secpos\*STM\_SECTOR\_SIZE+STM32\_FLASH\_BASE);//擦除这个扇区

for(i=0;i<secremain;i++)//复制

{

STMFLASH\_BUF[i+secoff]=pBuffer[i];

}

STMFLASH\_Write\_NoCheck(secpos\*STM\_SECTOR\_SIZE+STM32\_FLASH\_BASE,STMFLASH\_BUF,STM\_SECTOR\_SIZE/2);//写入整个扇区

}else STMFLASH\_Write\_NoCheck(WriteAddr,pBuffer,secremain);//写已经擦除了的,直接写入扇区剩余区间.

if(NumToWrite==secremain)break;//写入结束了

else//写入未结束

{

secpos++; //扇区地址增1

secoff=0; //偏移位置为0

pBuffer+=secremain; //指针偏移

WriteAddr+=secremain; //写地址偏移

NumToWrite-=secremain; //字节(16位)数递减

if(NumToWrite>(STM\_SECTOR\_SIZE/2))secremain=STM\_SECTOR\_SIZE/2;//下一个扇区还是写不完

else secremain=NumToWrite;//下一个扇区可以写完了

}

};

FLASH\_Lock();//上锁

}

#endif

//从指定地址开始读出指定长度的数据

//ReadAddr:起始地址

//pBuffer:数据指针

//NumToWrite:半字(16位)数

void STMFLASH\_Read(u32 ReadAddr,u16 \*pBuffer,u16 NumToRead)

{

u16 i;

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

{

pBuffer[i]=STMFLASH\_ReadHalfWord(ReadAddr);//读取2个字节.

ReadAddr+=2;//偏移2个字节.

}

}

//WriteAddr:起始地址

//WriteData:要写入的数据

void Test\_Write(u32 WriteAddr,u16 WriteData)

{

STMFLASH\_Write(WriteAddr,&WriteData,1);//写入一个字

}

# **系统原理图**

系统原理图设计如图 2‑1所示

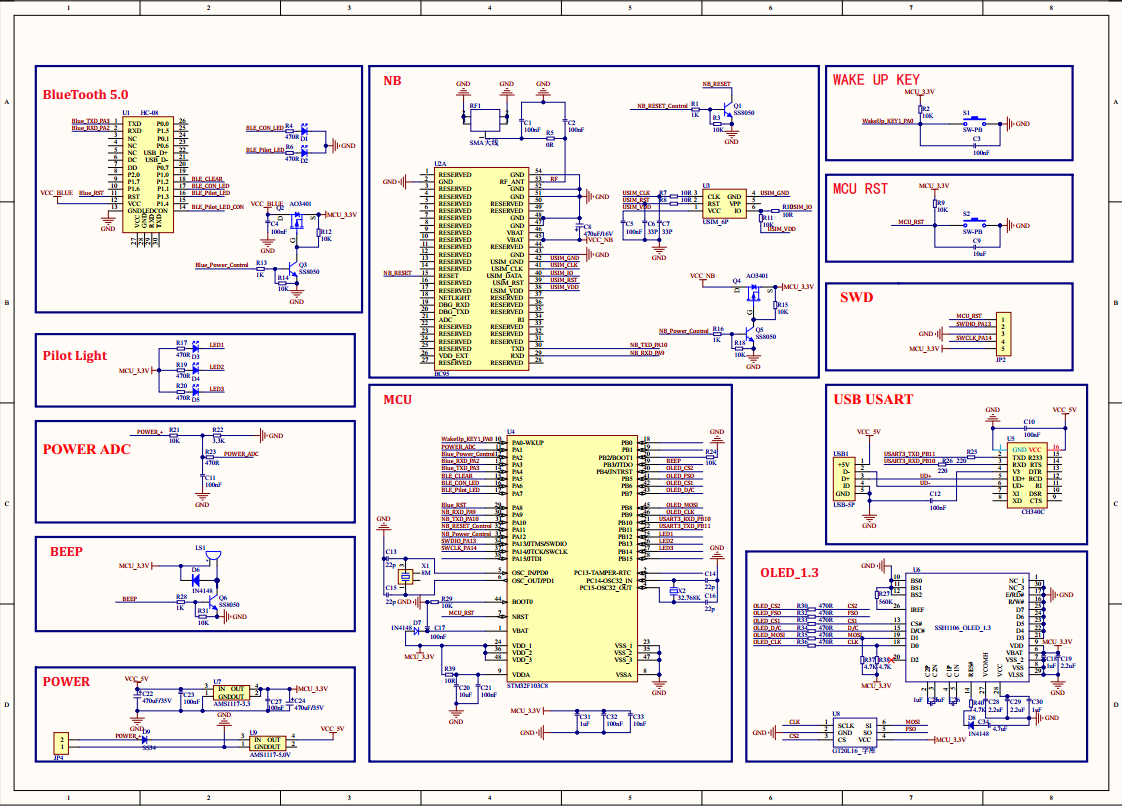


图 2‑1 图系统原理图

# 系统PCB板

系统PCB板设计图 3‑1如所示

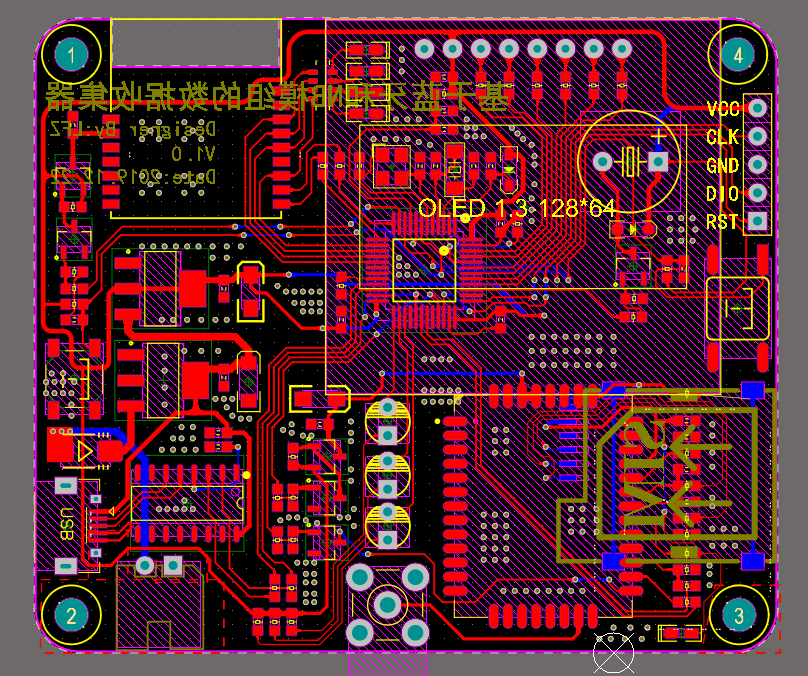


图 3‑1 系统PCB板图

# PCB 3D图

系统PCB板设计3D如图 4‑1所示

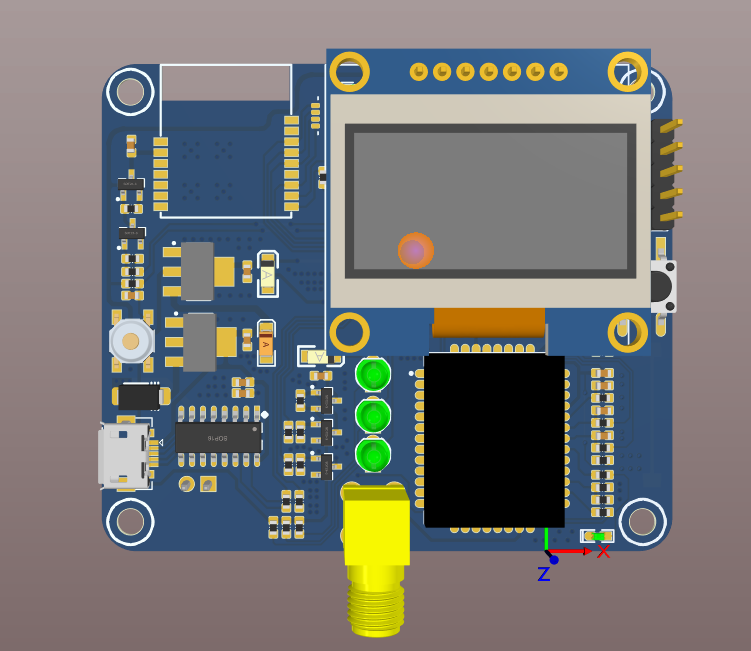


图 4‑1数据采集器PCB 3D图

# 实物图

收据采集器实物如图 5‑1所示

****

图 5‑1数据采集器实物图