#include "STC8G.h" //主控芯片的头文件

#include "intrins.h" //因为要用到nop()函数故而包含此文件

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*常用数据类型定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define u8 uint8\_t

#define u16 uint16\_t

#define u32 uint32\_t

typedef unsigned char uint8\_t;

typedef unsigned int uint16\_t;

typedef unsigned long uint32\_t;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*端口/引脚定义区域\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

sbit Mcode=P3^2; //曼彻斯特码输入端

sbit LCDRS=P3^6; //LCD1602数据/命令选择端口

sbit LCDEN=P3^7; //LCD1602使能信号端口

#define LCDDATA P1 //LCD1602数据端口（仅用DB4至DB7）

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*用户自定义数据区域\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

u8 table1[]="10b: ";//LCD1602显示10位卡号

u8 table2[]="08b: ";//LCD1602显示8位卡号

#define TIME\_OUT 256 //最大超时阈值

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*函数声明区域\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void delay(u16 Count); //延时函数

void IO\_init(void); //IO初始化函数

void SYSCLK\_CCO(u8 TYPE,u8 SET\_F,u8 SET\_P);//系统时钟输出函数

void LCD1602\_Write(u8 cmdordata,u8 writetype);

//写入液晶模组命令或数据函数

void LCD1602\_init(void); //LCD1602初始化函数

void LCD1602\_DIS(void); //显示字符函数

void LCD1602\_DIS\_CHAR(u8 x,u8 y,u8 z);//在设定地址写入字符数据函数

u32 Read\_Card(void); //读卡函数

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*主函数区域\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main(void)

{

u32 SN=0; //定义卡号接收变量

u16 a; //定义8位卡号的首部

u32 b; //定义8位卡号的尾部

IO\_init(); //IO初始化

SYSCLK\_CCO(1,96,1); //P1.6输出96分频的内部高速时钟12M/96=125kHz

LCDEN=1;delay(200); //使能置“1”（防止复位乱码）

LCDEN=0;delay(200); //使能清“0”（防止复位乱码）

LCD1602\_init(); //LCD1602初始化

delay(200); //延时等待稳定

LCD1602\_DIS(); //显示字符效果

while(1)

{

SN=Read\_Card(); //读取卡号数据

if(SN) //若卡号不为0，即有效卡号

{

//显示10位卡号

LCD1602\_DIS\_CHAR(1,5,SN/1000000000+0x30);

LCD1602\_DIS\_CHAR(1,6,SN%1000000000/100000000+0x30);

LCD1602\_DIS\_CHAR(1,7,SN%100000000/10000000+0x30);

LCD1602\_DIS\_CHAR(1,8,SN%10000000/1000000+0x30);

LCD1602\_DIS\_CHAR(1,9,SN%1000000/100000+0x30);

LCD1602\_DIS\_CHAR(1,10,SN%100000/10000+0x30);

LCD1602\_DIS\_CHAR(1,11,SN%10000/1000+0x30);

LCD1602\_DIS\_CHAR(1,12,SN%1000/100+0x30);

LCD1602\_DIS\_CHAR(1,13,SN%100/10+0x30);

LCD1602\_DIS\_CHAR(1,14,SN%10+0x30);

//显示8位卡号

/\*转换方法：10位卡号转换为十六进制数是4个字节, 第一个字节舍弃,

第二个字节转换为8位卡号的前3位，第三个和第四个字节一起转换为8

位卡号的后5位即可。例如：某卡片的十位卡号0319631142对应的十六

进制形式为130D2F26，先把第一个字节“13”舍弃，把第二个字节“0D”转

换为十进制就是13，第三个和第四个字节“2F26”转换为十进制就是12070，

所以该卡片的8位形式就是13，12070。\*/

a=(SN/16/16/16/16/16%16)\*16+(SN/16/16/16/16%16);

LCD1602\_DIS\_CHAR(2,5,a/100+0x30);

LCD1602\_DIS\_CHAR(2,6,a%100/10+0x30);

LCD1602\_DIS\_CHAR(2,7,a%10+0x30);

LCD1602\_DIS\_CHAR(2,8,',');

b=(SN/16/16/16%16)\*4096+(SN/16/16%16)\*256+(SN/16%16)\*16+(SN%16);

LCD1602\_DIS\_CHAR(2,9,b/10000+0x30);

LCD1602\_DIS\_CHAR(2,10,b%10000/1000+0x30);

LCD1602\_DIS\_CHAR(2,11,b%1000/100+0x30);

LCD1602\_DIS\_CHAR(2,12,b%100/10+0x30);

LCD1602\_DIS\_CHAR(2,13,b%10+0x30);

}

}

}

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

//延时函数delay()，有形参Count用于控制延时函数执行次数，无返回值

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

void delay(u16 Count)

{

u8 i,j;

while (Count--)

{

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

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

}

}

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

//384us延时函数delay384us()，无形参，无返回值，这是用STC-ISP算出

//来的384us延时，此延时可以不严格，但应大于256us小于512us

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

void delay384us(void)//@12MHz

{

u8 i, j;

\_nop\_();

\_nop\_();

i=6;

j=249;

do

{

while(--j);

} while(--i);

}

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

//IO初始化函数IO\_Init()，无形参，无返回值

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

void IO\_init(void)

{

//配置P1.6为推挽/强上拉模式

P1M0|=0x40; //P1M0.6=1

P1M1&=0xBF; //P1M1.6=0

//配置P3.2为准双向/弱上拉模式

P3M0&=0xFB; //P3M0.2=0

P3M1&=0xFB; //P3M1.2=0

//配置P3.6-7为准双向/弱上拉模式

P3M0&=0x3F; //P3M0.6-7=0

P3M1&=0x3F; //P3M1.6-7=0

//配置P1.0-3为准双向/弱上拉模式

P1M0&=0xF0; //P1M0.0-3=0

P1M1&=0xF0; //P1M1.0-3=0

delay(10); //等待I/O模式配置稳定

}

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

//系统时钟输出函数SYSCLK\_CCO()，有形参TYPE用于选择时钟源（0-外部时钟）

//（1-片内高速时钟）（2-片内低速时钟），有形参SET\_F用于指定时钟源

//分频系数，有形参SET\_P用于指定输出引脚的选择（P5.4/P1.6）无返回值

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

void SYSCLK\_CCO(u8 TYPE,u8 SET\_F,u8 SET\_P)

{

P\_SW2|=0x80; //允许访问扩展特殊功能寄存器XSFR

switch(TYPE)

{

case 0:

{

XOSCCR|=0xC0; //使能外部晶体振荡器

while(!(XOSCCR&0x01));//等待外部时钟稳定

//配置MCKSEL[1:0]=“01”，选择外部时钟

CLKSEL&=0xFD; //清零MCKSEL[1:0]的高位

CLKSEL|=0x01; //配置MCKSEL[1:0]的低位

}break;

case 1: //若选择片内高速时钟，则无需配置时钟选择

{}break;

case 2:

{

IRC32KCR|=0x80; //使能内部32kHz低速IRC控制寄存器

while(!(IRC32KCR&0x01));//等待内部低速时钟稳定

CLKSEL|=0x03; //配置MCKSEL[1:0]=“11”选择内部低速IRC时钟

}break;

}

switch(SET\_F)

{

case 1:{MCLKOCR&=0x80;MCLKOCR|=0x01;}break; //Fsysclk/1

case 2:{MCLKOCR&=0x80;MCLKOCR|=0x02;}break; //Fsysclk/2

case 4:{MCLKOCR&=0x80;MCLKOCR|=0x04;}break; //Fsysclk/4

case 8:{MCLKOCR&=0x80;MCLKOCR|=0x08;}break; //Fsysclk/8

case 16:{MCLKOCR&=0x80;MCLKOCR|=0x10;}break; //Fsysclk/16

case 32:{MCLKOCR&=0x80;MCLKOCR|=0x20;}break; //Fsysclk/32

case 64:{MCLKOCR&=0x80;MCLKOCR|=0x40;}break; //Fsysclk/64

case 96:{MCLKOCR&=0x80;MCLKOCR|=0x60;}break; //Fsysclk/96

case 127:{MCLKOCR&=0x80;MCLKOCR|=0x7F;}break; //Fsysclk/127

}

if(SET\_P==0)

MCLKOCR&=0x7F; //配置时钟由P5.4引脚输出

else

MCLKOCR|=0x80; //配置时钟由P1.6引脚输出

P\_SW2&=0x7F; //结束并关闭XSFR访问

}

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

//LCD1602初始化函数LCD1602\_init()，无形参和返回值

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

void LCD1602\_init(void)

{

LCD1602\_Write(0x28,0);delay(10);//数据总线为4位，显示2行，5\*7点阵/每字符

LCDEN=1;delay(10); //使能置“1”

LCDEN=0;delay(10); //使能清“0”

LCD1602\_Write(0x0C,0);delay(10);//设置开显示，不显示光标

LCD1602\_Write(0x06,0);delay(10);//写入新数据后显示屏整体不移动仅光标右移

LCD1602\_Write(0x01,0);delay(10);//写入清屏命令

}

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

//显示字符函数LCD1602\_DIS()，无形参和返回值

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

void LCD1602\_DIS(void)

{

u8 i; //定义控制循环变量i

LCD1602\_Write(0x80,0);//选择第一行

delay(5); //延时等待1602稳定

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

{

LCD1602\_Write(table1[i],1);//写入table1[]内容

delay(1);

}

LCD1602\_Write(0xC0,0);//选择第二行

delay(5); //延时等待1602稳定

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

{

LCD1602\_Write(table2[i],1);//写入table2[]内容

delay(1);

}

}

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

//写入液晶模组命令或数据函数LCD1602\_Write()，有形参cmdordata

//和writetype，无返回值

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

void LCD1602\_Write(u8 cmdordata,u8 writetype)

{

u8 temp=0; //定义数据处理中间变量

LCDRS=writetype; //判断写入类型0为命令1为数据

delay(2); //延时等待稳定

LCDDATA&=0xF0; //清数据引脚

temp=cmdordata>>4; //高四位移到低四位

LCDDATA|=temp&0x0F; //写高四位

LCDEN=1;delay(2); //使能置“1”

LCDEN=0;delay(2); //使能清“0”

LCDDATA&=0xF0; //清数据引脚

LCDDATA|=cmdordata&0x0F; //写低四位

LCDEN=1;delay(2); //使能置“1”

LCDEN=0;delay(2); //使能清“0”

}

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

//设定地址写入字符函数LCD1602\_DIS\_CHAR()，有形参x、y、z无返回值

//x表示1602液晶的行，y表示列地址，z表示欲写入的字符

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

void LCD1602\_DIS\_CHAR(u8 x,u8 y,u8 z)

{

u8 address;

if(x==1) //若欲显示在第一行

address=0x80+y; //第一行的行首地址+列地址

else

address=0xC0+y; //第二行的行首地址+列地址

LCD1602\_Write(address,0); //设定显示地址

LCD1602\_Write(z,1); //写入字符数据

}

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

//读卡函数Read\_Card()，无形参，有返回值temp

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

u32 Read\_Card(void)

{

u8 i=0; //起始位循环控制变量

u8 timeout; //超时计数变量

u8 timeout\_flag; //超时标志

u8 row,col; //行列变量

u8 row\_parity; //行校验变量

u8 col\_parity[5]; //列校验变量

u8 M\_data; //数据位变量

u32 temp; //卡号变量

u32 search=0; //搜索次数变量

while(1) //进行完整的读卡过程

{

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

if(search==10) //若10次搜索都未发现有效数据帧

return 0; //则返回0

else //否则

search++; //搜索次数值自增

timeout=0; //超时值清零

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

while(Mcode==0) //等待接收引脚出现低电平

{

if(timeout==TIME\_OUT) //若超时值等于设定阈值

break; //直接跳出本次搜索

else //若超时值小于设定阈值

timeout++; //超时值自增

}

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

if(timeout==100) //若超时值等于100

continue; //结束本次搜索

else //否则

timeout=0; //超时值清零

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

delay384us(); //延时384us

if(Mcode) //开始寻找帧头起始位

{

for(i=0;i<8;i++) //循环8次判断和处理过程

{

timeout=0; //先把超时值清零

while(Mcode) //等待接收引脚为高电平

{

if(timeout==TIME\_OUT) //若超时值等于设定阈值

{

timeout\_flag=1;break; //判定为超时事件，直接退出

}

else timeout++; //否则超时值自增

}

delay384us(); //再延时384us，判断下一个位

if(Mcode&&timeout\_flag==0);

//判断下一位是否为1且超时标指是否为0

else //两个条件中若有其一不满足

break; //则直接退出

}

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

if(timeout\_flag) //若超时标志为1

{

timeout\_flag=0;continue;//则清零标志并退出本次循环

}

else;

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

if(i==8)//若i=8，则说明帧头已被识别，现在开始接受数据

{

timeout\_flag=0; //清零超时标志

timeout=0; //清零超时值

while(Mcode) //等待接收引脚为高电平

{

if(timeout==TIME\_OUT) //若超时值等于设定阈值

{

timeout\_flag=0;break; //判定为超时事件，直接退出

}

else timeout++; //否则超时值自增

}

if(timeout\_flag) //若超时标志为1

{

timeout\_flag=0;continue;//则清零标志并退出本次循环

}

else;

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

//将5个列校验值清零

col\_parity[0]=col\_parity[1]=col\_parity[2]=0;

col\_parity[3]=col\_parity[4]=0;

for(row=0;row<11;row++)

//循环11次，接收11行数据（4个数据位+1个行校验值）

{

for(col=0,row\_parity=0;col<5;col++)//接收数据每5个为一行

{

delay384us(); //延时384us，判断下一个数据位

if(Mcode)

M\_data=1; //若接收引脚高电平，则数据位为1

else

M\_data=0; //否则，数据位为0

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

if(col<4&&row<10) //若接收数据位在0-3列且在1-10行内

//意思就是在有效数据区域内

{

temp<<=1; //卡号变量左移1位

temp+=(u32)M\_data;//把新的数据位加到卡号变量中

}

else;

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

row\_parity+=M\_data; //将数据加到行校验变量

col\_parity[col]+=M\_data;//将数据加到列校验变量

//虽然第5列数据仅作校验，但也加上

timeout=0; //清零超时值

while(Mcode==(bit)M\_data) //等待接收引脚状态为停止位

{

if(timeout==TIME\_OUT) //若超时值等于设定阈值

{

timeout\_flag=1;break; //判定为超时事件，直接退出

}

else timeout++; //否则超时值自增

}

if(timeout\_flag) //若超时标志为1

break; //则退出本次循环

else;

}

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

if(row<10) //因为最后一行没有校验

//所以要加限制

{

if((row\_parity&0x01)||timeout\_flag)

//若校验出错或超时标识为1

{

temp=0; //清零本次采集的卡号数据

timeout\_flag=1; //置位超时标志位

break; //退出

}

else;

}

else;

}

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

//对最后接收的列校验进行判断并结束本次主循环

if(timeout\_flag||((col\_parity[0]&0x01)&&\

(col\_parity[1]&0x01)&&(col\_parity[2]&0x01)&&\

(col\_parity[3]&0x01)))

{ //最后一列是没有校验的

timeout\_flag=0; //清零超时标志位

temp=0; //清零本次采集的卡号数据

continue; //退出本次循环

}

else

return temp; //将卡号数据返回

}

continue; //退出本次循环

}

continue; //退出本次循环

}

}