#include "reg52.h"

#include "main.h"

#include "uart.h"

#include "timer.h"

#include "rc522.h"

#include "ctrl.h"

#include "lcd12864.h"

#include "beep.h"

#include "key.h"

#include "EEPROM.h"

void init\_all(void) //初始化

{

EA = 0; //关总中断

init\_timer(); //定时器初始化

init\_uart(); //串口初始化

init\_rc522(); //RC522初始化

lcd\_init(); //LCD初始化

EEPROM\_READ(0,Card\_SN\_BUF,4);//EEPROM初始化,取出记录的卡号

EEPROM\_READ(7,KEY\_BUF,8);//EEPROM初始化,取出记录的密码

send\_bytes(Card\_SN\_BUF,4);

EA = 1; //开总中断

}

void main(void) //主函数

{

INT8U key;

Delay\_ms(50); //让硬件稳定

init\_all(); //执行初始化函数

relay\_OFF(); //关继电器

LED\_BLINK\_1(); //led test

beep1(); //beep test

display(0,0,0); //显示初始化

while(1)

{

key=key\_scan(); //按键操作

if(key==12) if(states>0) states--;

else states=0; //上一功能

if(key==13) if(++states>3) states=3; //下一功能

ctrl\_process(); //进入RC522操作

}

}

#include "reg52.h"

#include "string.h"

#include "main.h"

#include "uart.h"

#include "rc522.h"

#include "ctrl.h"

#include "beep.h"

#include "lcd12864.h"

#include "timer.h"

#include "EEPROM.h"

#include "key.h"

INT8U states=0;//状态机

INT8U Card\_type[2], //卡片类型

Card\_SN[4]={0},//IC卡号

Card\_SN\_BUF[4]={0xff,0xff,0xff,0xff},//卡号登记缓冲区

KEY\_BUF[8]={'0','0','0','0','0','0','0','0'};

INT8U bPass = 0; bWarn = 0;

void uart\_over( void ) //串口数据还原

{

UartCount = 0;

UartStart = FALSE;

UartComp = FALSE;

}

void pass( void )

{

beep1();

Delay\_ms(2000);//让卡片离开

LED\_BLINK\_1();

}

void warn(void)

{

LED\_BLINK\_2();

}

INT8U IC\_READ( void )

{

INT8U ID\_ASC[8],i;

if( PcdRequest( PICC\_REQIDL, Card\_type ) != MI\_OK )//寻天线区内未进入休眠状态的卡，返回卡片类型 2字节

{

if( PcdRequest( PICC\_REQIDL, Card\_type ) != MI\_OK )//寻天线区内未进入休眠状态的卡，返回卡片类型 2字节

{

return FALSE;

}

}

if( PcdAnticoll( Card\_SN ) != MI\_OK ) //防冲撞，返回卡的序列号 4字节

{

bWarn = 1;

return FALSE;

}

bPass = 1;

send\_bytes(Card\_type,2);

send\_bytes(Card\_SN,4);

for(i=0;i<4;i++) //卡ID号转化成ASCII

{

if(Card\_SN[i]/16>9) ID\_ASC[i\*2]=Card\_SN[i]/16+'7';

else ID\_ASC[i\*2]=Card\_SN[i]/16+'0';

if(Card\_SN[i]%16>9) ID\_ASC[i\*2+1]=Card\_SN[i]%16+'7';

else ID\_ASC[i\*2+1]=Card\_SN[i]%16+'0';

}

display2(3,0,"ID: ",4); //显示卡号

display2(3,2,ID\_ASC,8);

return TRUE;

}

void ctrl\_process( void )

{

INT8U i,key\_count,key\_value=16,table[8]="--------",statesbuf;

if(states!=statesbuf) //状态改变，清屏

{

display\_clear\_line(2);

display\_clear\_line(3);

statesbuf=states;

}

switch(states)

{

case 0: //IC卡读卡输入

display(1,0,1);

if(IC\_READ())

{

if ((Card\_SN[0]==Card\_SN\_BUF[0])

&&(Card\_SN[1]==Card\_SN\_BUF[1])

&&(Card\_SN[2]==Card\_SN\_BUF[2])

&&(Card\_SN[3]==Card\_SN\_BUF[3]))

{

relay\_ON();//灯开关

display(2,0,5);

}

else display(2,0,6);

relay\_OFF();

}

break;

case 1:

display(1,0,2); //密码输入

display2(3,0,table,8);

key\_count=0;

while(1)

{

key\_value=key\_scan();

if(key\_value==12)

{

states--;

return;

}

if(key\_value==13)

{

states++;

return;

}

if(key\_value>=0 && key\_value <=9)//有按键输入

{

table[key\_count++]=key\_value+'0';

display2(3,0,table,8);

}

if(key\_value==11)//退格

{

table[--key\_count]='-';

display2(3,0,table,8);

}

if(key\_count==8)

{

if( table[0]==KEY\_BUF[0] &&

table[1]==KEY\_BUF[1] &&

table[2]==KEY\_BUF[2] &&

table[3]==KEY\_BUF[3] &&

table[4]==KEY\_BUF[4] &&

table[5]==KEY\_BUF[5] &&

table[6]==KEY\_BUF[6] &&

table[7]==KEY\_BUF[7] ) //密吗正确

{

bPass=1;

relay\_ON();//灯开关

display(2,0,5);

relay\_OFF();

break;

}

else //密码错误

{

relay\_OFF();

beep1() ;

bWarn=1;

display(2,0,6);

break;

}

}

}

break ;

case 2:

display(1,0,3); //IC卡登记

if(IC\_READ())

{

for(i=0;i<4;i++)

Card\_SN\_BUF[i] = Card\_SN[i];

EEPROM\_WRITE(0,Card\_SN,4);//写入EEPROM

}

break ;

case 3:

display(1,0,4); //密码设置

display2(3,0,table,8);

key\_count=0;

while(1)

{

key\_value=key\_scan();

if(key\_value==12)

{

states--;

return;

}

if(key\_value>=0 && key\_value <=9)//有按键输入

{

table[key\_count++]=key\_value+'0';

display2(3,0,table,8);

}

if(key\_value==11)//退格

{

table[--key\_count]='-';

display2(3,0,table,8);

}

if(key\_count==8 && key\_value == 15) //按下确定键

{

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

KEY\_BUF[i]=table[i];

EEPROM\_WRITE(7,KEY\_BUF,8);//写入EEPROM

break;

}

}

break ;

default : break;

}

if( bPass ) //处理成功

{

bPass = 0;

pass();

}

if( bWarn ) //处理失败

{

bWarn = 0;

warn();

}

}

#include "reg52.h"

#include "intrins.h"

#include "main.h"

#include "rc522.h"

#include "string.h"

void delay\_ns(unsigned int ns)

{

unsigned int i;

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

{

nop();

nop();

nop();

}

}

//------------------------------------------

// 读SPI数据

//------------------------------------------

unsigned char SPIReadByte(void)

{

unsigned char SPICount; // Counter used to clock out the data

unsigned char SPIData;

SPIData = 0;

for (SPICount = 0; SPICount < 8; SPICount++) // Prepare to clock in the data to be read

{

SPIData <<=1; // Rotate the data

CLR\_SPI\_CK; //nop();//nop(); // Raise the clock to clock the data out of the MAX7456

if(STU\_SPI\_MISO)

{

SPIData|=0x01;

}

SET\_SPI\_CK; //nop();//nop(); // Drop the clock ready for the next bit

} // and loop back

return (SPIData); // Finally return the read data

}

//------------------------------------------

// 写SPI数据

//------------------------------------------

void SPIWriteByte(unsigned char SPIData)

{

unsigned char SPICount; // Counter used to clock out the data

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

{

if (SPIData & 0x80)

{

SET\_SPI\_MOSI;

}

else

{

CLR\_SPI\_MOSI;

}

nop();nop();

CLR\_SPI\_CK;nop();nop();

SET\_SPI\_CK;nop();nop();

SPIData <<= 1;

}

}

/////////////////////////////////////////////////////////////////////

//功 能：读RC632寄存器

//参数说明：Address[IN]:寄存器地址

//返 回：读出的值

/////////////////////////////////////////////////////////////////////

unsigned char ReadRawRC(unsigned char Address)

{

unsigned char ucAddr;

unsigned char ucResult=0;

CLR\_SPI\_CS;

ucAddr = ((Address<<1)&0x7E)|0x80;

SPIWriteByte(ucAddr);

ucResult=SPIReadByte();

SET\_SPI\_CS;

return ucResult;

}

/////////////////////////////////////////////////////////////////////

//功 能：写RC632寄存器

//参数说明：Address[IN]:寄存器地址

// value[IN]:写入的值

/////////////////////////////////////////////////////////////////////

void WriteRawRC(unsigned char Address, unsigned char value)

{

unsigned char ucAddr;

CLR\_SPI\_CS;

ucAddr = ((Address<<1)&0x7E);

SPIWriteByte(ucAddr);

SPIWriteByte(value);

SET\_SPI\_CS;

}

/////////////////////////////////////////////////////////////////////

//功 能：清RC522寄存器位

//参数说明：reg[IN]:寄存器地址

// mask[IN]:清位值

/////////////////////////////////////////////////////////////////////

void ClearBitMask(unsigned char reg,unsigned char mask)

{

char tmp = 0x00;

tmp = ReadRawRC(reg);

WriteRawRC(reg, tmp & ~mask); // clear bit mask

}

/////////////////////////////////////////////////////////////////////

//功 能：置RC522寄存器位

//参数说明：reg[IN]:寄存器地址

// mask[IN]:置位值

/////////////////////////////////////////////////////////////////////

void SetBitMask(unsigned char reg,unsigned char mask)

{

char tmp = 0x00;

tmp = ReadRawRC(reg);

WriteRawRC(reg,tmp | mask); // set bit mask

}

/////////////////////////////////////////////////////////////////////

//用MF522计算CRC16函数

/////////////////////////////////////////////////////////////////////

void CalulateCRC(unsigned char \*pIndata,unsigned char len,unsigned char \*pOutData)

{

unsigned char i,n;

ClearBitMask(DivIrqReg,0x04);

WriteRawRC(CommandReg,PCD\_IDLE);

SetBitMask(FIFOLevelReg,0x80);

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

{ WriteRawRC(FIFODataReg, \*(pIndata+i)); }

WriteRawRC(CommandReg, PCD\_CALCCRC);

i = 0xFF;

do

{

n = ReadRawRC(DivIrqReg);

i--;

}

while ((i!=0) && !(n&0x04));

pOutData[0] = ReadRawRC(CRCResultRegL);

pOutData[1] = ReadRawRC(CRCResultRegM);

}

/////////////////////////////////////////////////////////////////////

//功 能：通过RC522和ISO14443卡通讯

//参数说明：Command[IN]:RC522命令字

// pInData[IN]:通过RC522发送到卡片的数据

// InLenByte[IN]:发送数据的字节长度

// pOutData[OUT]:接收到的卡片返回数据

// \*pOutLenBit[OUT]:返回数据的位长度

/////////////////////////////////////////////////////////////////////

char PcdComMF522(unsigned char Command,

unsigned char \*pInData,

unsigned char InLenByte,

unsigned char \*pOutData,

unsigned int \*pOutLenBit)

{

char status = MI\_ERR;

unsigned char irqEn = 0x00;

unsigned char waitFor = 0x00;

unsigned char lastBits;

unsigned char n;

unsigned int i;

switch (Command)

{

case PCD\_AUTHENT:

irqEn = 0x12;

waitFor = 0x10;

break;

case PCD\_TRANSCEIVE:

irqEn = 0x77;

waitFor = 0x30;

break;

default:

break;

}

WriteRawRC(ComIEnReg,irqEn|0x80);

ClearBitMask(ComIrqReg,0x80);

WriteRawRC(CommandReg,PCD\_IDLE);

SetBitMask(FIFOLevelReg,0x80);

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

{

WriteRawRC(FIFODataReg, pInData[i]);

}

WriteRawRC(CommandReg, Command);

if (Command == PCD\_TRANSCEIVE)

{

SetBitMask(BitFramingReg,0x80);

}

//i = 600;//根据时钟频率调整，操作M1卡最大等待时间25ms

i = 2000;

do

{

n = ReadRawRC(ComIrqReg);

i--;

}

while ((i!=0) && !(n&0x01) && !(n&waitFor));

ClearBitMask(BitFramingReg,0x80);

if (i!=0)

{

if(!(ReadRawRC(ErrorReg)&0x1B))

{

status = MI\_OK;

if (n & irqEn & 0x01)

{ status = MI\_NOTAGERR; }

if (Command == PCD\_TRANSCEIVE)

{

n = ReadRawRC(FIFOLevelReg);

lastBits = ReadRawRC(ControlReg) & 0x07;

if (lastBits)

{

\*pOutLenBit = (n-1)\*8 + lastBits;

}

else

{

\*pOutLenBit = n\*8;

}

if (n == 0)

{

n = 1;

}

if (n > MAXRLEN)

{

n = MAXRLEN;

}

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

{

pOutData[i] = ReadRawRC(FIFODataReg);

}

}

}

else

{

status = MI\_ERR;

}

}

SetBitMask(ControlReg,0x80); // stop timer now

WriteRawRC(CommandReg,PCD\_IDLE);

return status;

}

/////////////////////////////////////////////////////////////////////

//功 能：寻卡

//参数说明: req\_code[IN]:寻卡方式

// 0x52 = 寻感应区内所有符合14443A标准的卡

// 0x26 = 寻未进入休眠状态的卡

// pTagType[OUT]：卡片类型代码

// 0x4400 = Mifare\_UltraLight

// 0x0400 = Mifare\_One(S50)

// 0x0200 = Mifare\_One(S70)

// 0x0800 = Mifare\_Pro(X)

// 0x4403 = Mifare\_DESFire

//返 回: 成功返回MI\_OK

/////////////////////////////////////////////////////////////////////

char PcdRequest(unsigned char req\_code,unsigned char \*pTagType)

{

char status;

unsigned int unLen;

unsigned char ucComMF522Buf[MAXRLEN];

ClearBitMask(Status2Reg,0x08);

WriteRawRC(BitFramingReg,0x07);

SetBitMask(TxControlReg,0x03);

ucComMF522Buf[0] = req\_code;

status = PcdComMF522(PCD\_TRANSCEIVE,ucComMF522Buf,1,ucComMF522Buf,&unLen);

if ((status == MI\_OK) && (unLen == 0x10))

{

\*pTagType = ucComMF522Buf[0];

\*(pTagType+1) = ucComMF522Buf[1];

}

else

{

status = MI\_ERR;

}

return status;

}

/////////////////////////////////////////////////////////////////////

//功 能：防冲撞

//参数说明: pSnr[OUT]:卡片序列号，4字节

//返 回: 成功返回MI\_OK

/////////////////////////////////////////////////////////////////////

char PcdAnticoll(unsigned char \*pSnr)

{

char status;

unsigned char i,snr\_check=0;

unsigned int unLen;

unsigned char ucComMF522Buf[MAXRLEN];

ClearBitMask(Status2Reg,0x08);

WriteRawRC(BitFramingReg,0x00);

ClearBitMask(CollReg,0x80);

ucComMF522Buf[0] = PICC\_ANTICOLL1;

ucComMF522Buf[1] = 0x20;

status = PcdComMF522(PCD\_TRANSCEIVE,ucComMF522Buf,2,ucComMF522Buf,&unLen);

if (status == MI\_OK)

{

for (i=0; i<4; i++)

{

\*(pSnr+i) = ucComMF522Buf[i];

snr\_check ^= ucComMF522Buf[i];

}

if (snr\_check != ucComMF522Buf[i])

{

status = MI\_ERR;

}

}

SetBitMask(CollReg,0x80);

return status;

}

/////////////////////////////////////////////////////////////////////

//功 能：选定卡片

//参数说明: pSnr[IN]:卡片序列号，4字节

//返 回: 成功返回MI\_OK

/////////////////////////////////////////////////////////////////////

char PcdSelect(unsigned char \*pSnr)

{

char status;

unsigned char i;

unsigned int unLen;

unsigned char ucComMF522Buf[MAXRLEN];

ucComMF522Buf[0] = PICC\_ANTICOLL1;

ucComMF522Buf[1] = 0x70;

ucComMF522Buf[6] = 0;

for (i=0; i<4; i++)

{

ucComMF522Buf[i+2] = \*(pSnr+i);

ucComMF522Buf[6] ^= \*(pSnr+i);

}

CalulateCRC(ucComMF522Buf,7,&ucComMF522Buf[7]);

ClearBitMask(Status2Reg,0x08);

status = PcdComMF522(PCD\_TRANSCEIVE,ucComMF522Buf,9,ucComMF522Buf,&unLen);

if ((status == MI\_OK) && (unLen == 0x18))

{

status = MI\_OK;

}

else

{

status = MI\_ERR;

}

return status;

}

/////////////////////////////////////////////////////////////////////

//功 能：验证卡片密码

//参数说明: auth\_mode[IN]: 密码验证模式

// 0x60 = 验证A密钥

// 0x61 = 验证B密钥

// addr[IN]：块地址

// pKey[IN]：密码

// pSnr[IN]：卡片序列号，4字节

//返 回: 成功返回MI\_OK

/////////////////////////////////////////////////////////////////////

char PcdAuthState(unsigned char auth\_mode,unsigned char addr,unsigned char \*pKey,unsigned char \*pSnr)

{

char status;

unsigned int unLen;

unsigned char ucComMF522Buf[MAXRLEN];

ucComMF522Buf[0] = auth\_mode;

ucComMF522Buf[1] = addr;

memcpy(&ucComMF522Buf[2], pKey, 6);

memcpy(&ucComMF522Buf[8], pSnr, 6);

status = PcdComMF522(PCD\_AUTHENT,ucComMF522Buf,12,ucComMF522Buf,&unLen);

if ((status != MI\_OK) || (!(ReadRawRC(Status2Reg) & 0x08)))

{

status = MI\_ERR;

}

return status;

}

/////////////////////////////////////////////////////////////////////

//功 能：读取M1卡一块数据

//参数说明: addr[IN]：块地址

// pData[OUT]：读出的数据，16字节

//返 回: 成功返回MI\_OK

/////////////////////////////////////////////////////////////////////

char PcdRead(unsigned char addr,unsigned char \*pData)

{

char status;

unsigned int unLen;

unsigned char ucComMF522Buf[MAXRLEN];

ucComMF522Buf[0] = PICC\_READ;

ucComMF522Buf[1] = addr;

CalulateCRC(ucComMF522Buf,2,&ucComMF522Buf[2]);

status = PcdComMF522(PCD\_TRANSCEIVE,ucComMF522Buf,4,ucComMF522Buf,&unLen);

if ((status == MI\_OK) && (unLen == 0x90))

{

memcpy(pData, ucComMF522Buf, 16);

}

else

{

status = MI\_ERR;

}

return status;

}

/////////////////////////////////////////////////////////////////////

//功 能：写数据到M1卡一块

//参数说明: addr[IN]：块地址

// pData[IN]：写入的数据，16字节

//返 回: 成功返回MI\_OK

/////////////////////////////////////////////////////////////////////

char PcdWrite(unsigned char addr,unsigned char \*pData)

{

char status;

unsigned int unLen;

unsigned char ucComMF522Buf[MAXRLEN];

ucComMF522Buf[0] = PICC\_WRITE;

ucComMF522Buf[1] = addr;

CalulateCRC(ucComMF522Buf,2,&ucComMF522Buf[2]);

status = PcdComMF522(PCD\_TRANSCEIVE,ucComMF522Buf,4,ucComMF522Buf,&unLen);

if ((status != MI\_OK) || (unLen != 4) || ((ucComMF522Buf[0] & 0x0F) != 0x0A))

{

status = MI\_ERR;

}

if (status == MI\_OK)

{

memcpy(ucComMF522Buf, pData, 16);

CalulateCRC(ucComMF522Buf,16,&ucComMF522Buf[16]);

status = PcdComMF522(PCD\_TRANSCEIVE,ucComMF522Buf,18,ucComMF522Buf,&unLen);

if ((status != MI\_OK) || (unLen != 4) || ((ucComMF522Buf[0] & 0x0F) != 0x0A))

{

status = MI\_ERR;

}

}

return status;

}

/////////////////////////////////////////////////////////////////////

//功 能：命令卡片进入休眠状态

//返 回: 成功返回MI\_OK

/////////////////////////////////////////////////////////////////////

char PcdHalt(void)

{

char status;

unsigned int unLen;

unsigned char ucComMF522Buf[MAXRLEN];

ucComMF522Buf[0] = PICC\_HALT;

ucComMF522Buf[1] = 0;

CalulateCRC(ucComMF522Buf,2,&ucComMF522Buf[2]);

status = PcdComMF522(PCD\_TRANSCEIVE,ucComMF522Buf,4,ucComMF522Buf,&unLen);

return MI\_OK;

}

/////////////////////////////////////////////////////////////////////

//功 能：复位RC522

//返 回: 成功返回MI\_OK

/////////////////////////////////////////////////////////////////////

char PcdReset(void)

{

SET\_RC522RST;

delay\_ns(10);

CLR\_RC522RST;

delay\_ns(10);

SET\_RC522RST;

delay\_ns(10);

WriteRawRC(CommandReg,PCD\_RESETPHASE);

delay\_ns(10);

WriteRawRC(ModeReg,0x3D); //和Mifare卡通讯，CRC初始值0x6363

WriteRawRC(TReloadRegL,30);

WriteRawRC(TReloadRegH,0);

WriteRawRC(TModeReg,0x8D);

WriteRawRC(TPrescalerReg,0x3E);

WriteRawRC(TxAutoReg,0x40);//必须要

return MI\_OK;

}

//////////////////////////////////////////////////////////////////////

//设置RC632的工作方式

//////////////////////////////////////////////////////////////////////

char M500PcdConfigISOType(unsigned char type)

{

if (type == 'A') //ISO14443\_A

{

ClearBitMask(Status2Reg,0x08);

WriteRawRC(ModeReg,0x3D);//3F

WriteRawRC(RxSelReg,0x86);//84

WriteRawRC(RFCfgReg,0x7F); //4F

WriteRawRC(TReloadRegL,30);//tmoLength);// TReloadVal = 'h6a =tmoLength(dec)

WriteRawRC(TReloadRegH,0);

WriteRawRC(TModeReg,0x8D);

WriteRawRC(TPrescalerReg,0x3E);

delay\_ns(1000);

PcdAntennaOn();

}

else{ return -1; }

return MI\_OK;

}

/////////////////////////////////////////////////////////////////////

//开启天线

//每次启动或关闭天险发射之间应至少有1ms的间隔

/////////////////////////////////////////////////////////////////////

void PcdAntennaOn(void)

{

unsigned char i;

i = ReadRawRC(TxControlReg);

if (!(i & 0x03))

{

SetBitMask(TxControlReg, 0x03);

}

}

/////////////////////////////////////////////////////////////////////

//关闭天线

/////////////////////////////////////////////////////////////////////

void PcdAntennaOff(void)

{

ClearBitMask(TxControlReg, 0x03);

}

void init\_rc522(void)

{

PcdReset();

PcdAntennaOff();

PcdAntennaOn();

M500PcdConfigISOType( 'A' );

}

/////////////////////////////////////////////////////////////////////

//功 能：扣款和充值

//参数说明: dd\_mode[IN]：命令字

// 0xC0 = 扣款

// 0xC1 = 充值

// addr[IN]：钱包地址

// pValue[IN]：4字节增(减)值，低位在前

//返 回: 成功返回MI\_OK

/////////////////////////////////////////////////////////////////////

/\*char PcdValue(unsigned char dd\_mode,unsigned char addr,unsigned char \*pValue)

{

char status;

unsigned int unLen;

unsigned char ucComMF522Buf[MAXRLEN];

//unsigned char i;

ucComMF522Buf[0] = dd\_mode;

ucComMF522Buf[1] = addr;

CalulateCRC(ucComMF522Buf,2,&ucComMF522Buf[2]);

status = PcdComMF522(PCD\_TRANSCEIVE,ucComMF522Buf,4,ucComMF522Buf,&unLen);

if ((status != MI\_OK) || (unLen != 4) || ((ucComMF522Buf[0] & 0x0F) != 0x0A))

{ status = MI\_ERR; }

if (status == MI\_OK)

{

memcpy(ucComMF522Buf, pValue, 4);

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

//{ ucComMF522Buf[i] = \*(pValue+i); }

CalulateCRC(ucComMF522Buf,4,&ucComMF522Buf[4]);

unLen = 0;

status = PcdComMF522(PCD\_TRANSCEIVE,ucComMF522Buf,6,ucComMF522Buf,&unLen);

if (status != MI\_ERR)

{ status = MI\_OK; }

}

if (status == MI\_OK)

{

ucComMF522Buf[0] = PICC\_TRANSFER;

ucComMF522Buf[1] = addr;

CalulateCRC(ucComMF522Buf,2,&ucComMF522Buf[2]);

status = PcdComMF522(PCD\_TRANSCEIVE,ucComMF522Buf,4,ucComMF522Buf,&unLen);

if ((status != MI\_OK) || (unLen != 4) || ((ucComMF522Buf[0] & 0x0F) != 0x0A))

{ status = MI\_ERR; }

}

return status;

}\*/

/////////////////////////////////////////////////////////////////////

//功 能：备份钱包

//参数说明: sourceaddr[IN]：源地址

// goaladdr[IN]：目标地址

//返 回: 成功返回MI\_OK

/////////////////////////////////////////////////////////////////////

/\*char PcdBakValue(unsigned char sourceaddr, unsigned char goaladdr)

{

char status;

unsigned int unLen;

unsigned char ucComMF522Buf[MAXRLEN];

ucComMF522Buf[0] = PICC\_RESTORE;

ucComMF522Buf[1] = sourceaddr;

CalulateCRC(ucComMF522Buf,2,&ucComMF522Buf[2]);

status = PcdComMF522(PCD\_TRANSCEIVE,ucComMF522Buf,4,ucComMF522Buf,&unLen);

if ((status != MI\_OK) || (unLen != 4) || ((ucComMF522Buf[0] & 0x0F) != 0x0A))

{ status = MI\_ERR; }

if (status == MI\_OK)

{

ucComMF522Buf[0] = 0;

ucComMF522Buf[1] = 0;

ucComMF522Buf[2] = 0;

ucComMF522Buf[3] = 0;

CalulateCRC(ucComMF522Buf,4,&ucComMF522Buf[4]);

status = PcdComMF522(PCD\_TRANSCEIVE,ucComMF522Buf,6,ucComMF522Buf,&unLen);

if (status != MI\_ERR)

{ status = MI\_OK; }

}

if (status != MI\_OK)

{ return MI\_ERR; }

ucComMF522Buf[0] = PICC\_TRANSFER;

ucComMF522Buf[1] = goaladdr;

CalulateCRC(ucComMF522Buf,2,&ucComMF522Buf[2]);

status = PcdComMF522(PCD\_TRANSCEIVE,ucComMF522Buf,4,ucComMF522Buf,&unLen);

if ((status != MI\_OK) || (unLen != 4) || ((ucComMF522Buf[0] & 0x0F) != 0x0A))

{ status = MI\_ERR; }

return status;

}\*/

#include <reg52.h>

#include <intrins.h>

#include "main.h"

#define uchar unsigned char

#define uint unsigned int

/\*LCD12864 端口定义\*/

#define LCD\_data P0 //数据口

sbit LCD\_RS = P2^5; //寄存器选择输入

sbit LCD\_RW = P2^6; //液晶读/写控制

sbit LCD\_EN = P2^7; //液晶使能控制

sbit LCM\_RST = P3^2;

//sbit LCD\_PSB = P0^4; //串/并方式控制,部分液晶不需要

uchar code dis0[]=" IC卡门禁系统 ";

uchar code dis1[]="1.IC卡输入 ";

uchar code dis2[]="2.密码输入:8位 ";

uchar code dis3[]="3.IC卡登记 ";

uchar code dis4[]="4.密码设置:8位 ";

uchar code dis5[]="成功 ";

uchar code dis6[]="失败 ";

void delay(int ms) //延时xms

{

while(ms--)

{

uchar i;

for(i=0;i<250;i++)

{

\_nop\_();

\_nop\_();

\_nop\_();

\_nop\_();

}

}

}

void delayNOP() //延时4us

{ \_nop\_(); \_nop\_(); \_nop\_(); \_nop\_();}

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

/\* \*/

/\*因为引脚的定义DB0..DB7是从P1.7...P1.0排列的 \*/

/\*所以顺序需要调换一下 \*/

/\* \*/

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

INT8U swapbit(INT8U udata)

{

INT8U ii,tmp=0;

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

{

tmp<<=1;

if(udata&0x01)

{

tmp|=0x01;

}

udata>>=1;

}

return tmp;

}

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

/\* \*/

/\*检查LCD忙状态 \*/

/\*lcd\_busy为1时，忙，等待。lcd-busy为0时,闲，可写指令与数据。 \*/

/\* \*/

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

bit lcd\_busy()

{

bit result;

LCD\_RS = 0;

LCD\_RW = 1;

LCD\_EN = 1;

delayNOP();

result = (bit)(LCD\_data&0x80);

LCD\_EN = 0;

return(result);

}

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

/\* \*/

/\*写指令数据到LCD \*/

/\*RS=L，RW=L，E=高脉冲，D0-D7=指令码。 \*/

/\* \*/

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

void write\_cmd(unsigned char WCLCM,BuysC)

{

unsigned int K=20000;

while(lcd\_busy() && K--); //预防没插LCD时候停在这

LCD\_RS = 0;

LCD\_RW = 0;

LCD\_EN = 0;

\_nop\_();

\_nop\_();

LCD\_data = swapbit(WCLCM);

delayNOP();

LCD\_EN = 1;

delayNOP();

LCD\_EN = 0;

}

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

/\* \*/

/\*写显示数据到LCD \*/

/\*RS=H，RW=L，E=高脉冲，D0-D7=数据。 \*/

/\* \*/

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

void lcd\_wdat(unsigned char WDLCM)

{

unsigned int K=20000;

while(lcd\_busy() && K--); //预防没插LCD时候停在这

LCD\_RS = 1;

LCD\_RW = 0;

LCD\_EN = 0;

LCD\_data= swapbit(WDLCM);

delayNOP();

LCD\_EN = 1;

delayNOP();

LCD\_EN = 0;

}

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

/\* \*/

/\* LCD初始化设定 \*/

/\* \*/

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

void lcd\_init()

{

// LCD\_PSB = 1; //并口方式 ，部分液晶不需要

// LCM\_RST =0;

write\_cmd(0x36); //扩充指令操作

delay(5);

write\_cmd(0x30); //基本指令操

delay(5);

write\_cmd(0x0C); //显示开，关光标

delay(5);

write\_cmd(0x01); //清除LCD的显示内容

delay(5);

}

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

/\* \*/

/\* 设定显示位置 \*/

/\* \*/

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

void lcd\_pos(uchar X,uchar Y)

{

uchar pos;

if (X==0)

{X=0x80;}

else if (X==1)

{X=0x90;}

else if (X==2)

{X=0x88;}

else if (X==3)

{X=0x98;}

pos = X+Y ;

write\_cmd(pos); //显示地址

}

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

/\* \*/

/\* 显示主程序 \*/

/\* \*/

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

void display\_all()

{

uchar i;

lcd\_pos(0,0); //第一行显示数字0~9

i = 0;

while(dis0[i] != '\0')

{

lcd\_wdat(dis0[i]); //显示字母

i++;

}

lcd\_pos(1,0); //第二行显示字母

i = 0;

while(dis1[i] != '\0')

{

lcd\_wdat(dis1[i]); //显示字母

i++;

}

lcd\_pos(2,0); //第三行显示文字

i = 0;

while(dis2[i] != '\0')

{

lcd\_wdat(dis2[i]);

i++;

}

lcd\_pos(3,0); //第四行显示广工无线电社

i = 0;

while(dis3[i] != '\0')

{

lcd\_wdat(dis3[i]);

i++;

}

}

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

/\* \*/

/\* 特定字符显示主程序 \*/

/\*

C:

0=" IC卡点台灯 ";

1="1.读卡中.... ";

2="2.登记.... ";

3="成功 ";

4="失败 ";

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

void display(uchar a,uchar b,uchar c)

{

uchar i;

lcd\_pos(a,b); //第一行显示数字0~9

i = 0;

switch(c)

{

case 0:

while(dis0[i] != '\0')

{

lcd\_wdat(dis0[i]); //显示字母

i++;

}

break;

case 1:

while(dis1[i] != '\0')

{

lcd\_wdat(dis1[i]); //显示字母

i++;

}

break;

case 2:

while(dis2[i] != '\0')

{

lcd\_wdat(dis2[i]); //显示字母

i++;

}

break;

case 3:

while(dis3[i] != '\0')

{

lcd\_wdat(dis3[i]); //显示字母

i++;

}

break;

case 4:

while(dis4[i] != '\0')

{

lcd\_wdat(dis4[i]); //显示字母

i++;

}

break;

case 5:

while(dis5[i] != '\0')

{

lcd\_wdat(dis5[i]); //显示字母

i++;

}

break;

case 6:

while(dis6[i] != '\0')

{

lcd\_wdat(dis6[i]); //显示字母

i++;

}

break;

default :break;

}

}

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

/\* \*/

/\* 一般字符显示主程序 \*/

/\*

\*p:内容

c:个数

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

void display2(uchar a,uchar b,uchar \*p,uchar num)

{

uchar i;

lcd\_pos(a,b); //第一行显示数字0~9

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

lcd\_wdat(p[i]); //显示字母

}

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

清除某一行

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

void display\_clear\_line(uchar a)

{

uchar i;

lcd\_pos(a,0);

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

lcd\_wdat(' ');

}

#include "reg52.h"

#include "timer.h"

#define uchar unsigned char

#define uint unsigned int

uchar num,temp;

uchar key\_scan(void) //键盘扫描

{

num=16;//16表示没按键输入

P1=0xfe; //扫描第一行

temp=P1;

temp&=0xf0;

if(temp!=0xf0) //检测有按键按下

{

Delay\_ms(5);

temp=P1;

temp&=0xf0;

if(temp!=0xf0)

{

temp=P1;

switch(temp) //判断按下是哪一个按键

{

case 0xee:num=1;

break;

case 0xde:num=2;

break;

case 0xbe:num=3;

break;

case 0x7e:num=12;

break;

}

while(temp!=0xf0)

{

temp=P1;

temp&=0xf0;

}

}

}

P1=0xfd; //扫描第二行

temp=P1;

temp&=0xf0;

if(temp!=0xf0)

{

Delay\_ms(5);

temp=P1;

temp&=0xf0;

if(temp!=0xf0)

{

temp=P1;

switch(temp)

{

case 0xed:num=4;

break;

case 0xdd:num=5;

break;

case 0xbd:num=6;

break;

case 0x7d:num=13;

break;

}

while(temp!=0xf0)

{

temp=P1;

temp&=0xf0;

}

}

}

P1=0xfb; //扫描第三行

temp=P1;

temp&=0xf0;

if(temp!=0xf0)

{

Delay\_ms(5);

temp=P1;

temp&=0xf0;

if(temp!=0xf0)

{

temp=P1;

switch(temp)

{

case 0xeb:num=7;

break;

case 0xdb:num=8;

break;

case 0xbb:num=9;

break;

case 0x7b:num=14;

break;

}

while(temp!=0xf0)

{

temp=P1;

temp&=0xf0;

}

}

}

P1=0xf7; //扫描第四行

temp=P1;

temp&=0xf0;

if(temp!=0xf0)

{

Delay\_ms(5);

temp=P1;

temp&=0xf0;

if(temp!=0xf0)

{

temp=P1;

switch(temp)

{

case 0xe7:num=0;

break;

case 0xd7:num=10;

break;

case 0xb7:num=11;

break;

case 0x77:num=15;

break;

}

while(temp!=0xf0)

{

temp=P1;

temp&=0xf0;

}

}

}

return num; //返回按键对应编号

}