物联网22-2王卫东

1598593280@qq.com

摘要

[通过迷人的摘要吸引您的读者。它通常是文件的简短摘要。   
当您准备好添加内容时，只需单击此处并开始键入。]

51单片机代码集

第一版

目录

[一、ADC模数转换 4](#_Toc131850568)

[**ADC.h** 4](#_Toc131850569)

[**ADC.c** 4](#_Toc131850570)

[二、AT24C02数据存储（EEPROM） 6](#_Toc131850571)

[**AT24C02.h** 6](#_Toc131850572)

[**AT24C02.c** 6](#_Toc131850573)

[三、DHT11温湿度传感器 8](#_Toc131850574)

[**DHT11.c** 8](#_Toc131850575)

[四、DS18B20温度传感器（单总线） 12](#_Toc131850576)

[**OneWire.h** 12](#_Toc131850577)

[**OneWire.c** 12](#_Toc131850578)

[**DS18B20.h** 14](#_Toc131850579)

[**DS18B20.c** 14](#_Toc131850580)

[五、DS1302时钟芯片 16](#_Toc131850581)

[**DS302.h** 16](#_Toc131850582)

[**DS1302.c** 16](#_Toc131850583)

[六、GY302光强传感器（I2C通信） 20](#_Toc131850584)

[**BH1750.h** 20](#_Toc131850585)

[七、I2C总线 25](#_Toc131850586)

[**I2C.h** 25](#_Toc131850587)

[八、LCD12864显示屏 28](#_Toc131850588)

[**lcd12864.h** 28](#_Toc131850589)

[**lcd12864.c** 28](#_Toc131850590)

[九、MP-4可燃气体检测 31](#_Toc131850591)

[**MP-4.c(示例)** 31](#_Toc131850592)

[十、PWM(示例) 32](#_Toc131850593)

[**breath\_light.h** 32](#_Toc131850594)

[**breath\_light.c** 32](#_Toc131850595)

[十一、RC522(RFID射频IC卡感应) 34](#_Toc131850596)

[**MAIN.H** 34](#_Toc131850597)

[**MAIN.C** 35](#_Toc131850598)

[**MFRC522.H** 38](#_Toc131850599)

[**RC522.c** 41](#_Toc131850600)

[十二、SYN6288语音合成 55](#_Toc131850601)

[**SYN6288.h** 55](#_Toc131850602)

[**SYN6288.c** 55](#_Toc131850603)

[**SYN6288\_voice.c(示例)** 56](#_Toc131850604)

[十三、步进电机 59](#_Toc131850605)

[**step\_motor.c** 59](#_Toc131850606)

[十四、蜂鸣器 61](#_Toc131850607)

[注意：普中A2型号的蜂鸣器引脚为P2^5 61](#_Toc131850608)

[**Buzzer.h** 61](#_Toc131850609)

[**Buzzer.c** 61](#_Toc131850610)

[十五、红外遥控 63](#_Toc131850611)

[**IR.h** 63](#_Toc131850612)

[**IR.c** 63](#_Toc131850613)

[十六、矩阵键盘 68](#_Toc131850614)

[**MatrixKey.h** 68](#_Toc131850615)

[**MatrixKey.c** 68](#_Toc131850616)

[十七、人体红外传感器 70](#_Toc131850617)

[**Body\_hw.c** 70](#_Toc131850618)

[十八、延时函数 71](#_Toc131850619)

[**delay.h** 71](#_Toc131850620)

[**delay.c** 71](#_Toc131850621)

[十九、中断、定时器 72](#_Toc131850622)

[**Timer0.h** 72](#_Toc131850623)

[**Timer1.h** 72](#_Toc131850624)

[**uart.h** 73](#_Toc131850625)

[二十、其他 75](#_Toc131850626)

[**main.h** 75](#_Toc131850627)

一、ADC模数转换

**ADC.h**

#ifndef \_ADC\_H

#define \_ADC\_H

//#include "main.h"

//#include "delay.h"

sbit ADC\_CLK = P3^6; //:定义XTP2046的时钟端口

sbit ADC\_CS = P3^5; //:定义XTP2046的使能端口

sbit ADC\_DIN = P3^4; //:定义XTP2046的串行数据输入口

sbit ADC\_DOUT = P3^7; //:定义XPT2046的串行数据输出口

u16 read\_AD\_dat(u8 cmd);

#endif

**ADC.c**

#include "ADC.h"

#include "intrins.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SPI写入数据\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void dat\_write(u8 dat)

{

u8 i;

ADC\_CLK = 0;

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

{

ADC\_DIN = dat >> 7; //:从最高位开始写入数据，右移7位读取的数据移入最低位

dat <<= 1; //:将最高位移走，次高位变为最高位，逐次读取数据

ADC\_CLK = 0; //:数据在时钟上升沿时写入

ADC\_CLK = 1;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SPI读入数据\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

u16 dat\_read()

{

u16 i,dat = 0;

ADC\_CLK = 0;

for (i = 0;i < 12;i++) //循环12次读入全部12位数据

{

dat <<= 1; //将读到的数据依次左移送入高位

ADC\_CLK = 1; //:下降沿读入数据

ADC\_CLK = 0;

dat |= ADC\_DOUT; //从最高位读入

}

return dat; //将读到的所有数据返回

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*读取AD模数转化数据\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

u16 read\_AD\_dat(u8 cmd)

{

u8 i;

u16 AD\_value;

ADC\_CLK = 0; //拉低使能端和时钟端口开始读取数据

ADC\_CS = 0;

dat\_write(cmd); //:指定检测通道

for(i = 6;i > 0;i--); //:依据时序图，等待BUSY拉低

ADC\_CLK = 1;

\_nop\_();

\_nop\_();

ADC\_CLK = 0;

\_nop\_();

\_nop\_();

AD\_value = dat\_read(); //:读取数字信号

ADC\_CS = 1; //:拉高使能端完成读取

return AD\_value;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*数据处理函数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void value\_deal(cmd)

{

u16 temp;

static u8 i;

if(i == 50)

{

i = 0;

temp = read\_AD\_dat(cmd);

}

i++;

}

二、AT24C02数据存储（EEPROM）

**AT24C02.h**

#ifndef \_\_AT24C02\_H\_\_

#define \_\_AT24C02\_H\_\_

void AT24C02\_WriteByte(unsigned char WordAddress,Data);

unsigned char AT24C02\_ReadByte(unsigned char WordAddress);

#endif

**AT24C02.c**

#include <REGX52.H>

#include "I2C.h"

#define AT24C02\_ADDRESS 0xA0

/\*\*

\* @brief AT24C02写入一个字节

\* @param WordAddress 要写入字节的地址

\* @param Data 要写入的数据

\* @retval 无

\*/

void AT24C02\_WriteByte(unsigned char WordAddress,Data)

{

I2C\_Start();

I2C\_SendByte(AT24C02\_ADDRESS);

I2C\_ReceiveAck();

I2C\_SendByte(WordAddress);

I2C\_ReceiveAck();

I2C\_SendByte(Data);

I2C\_ReceiveAck();

I2C\_Stop();

}

/\*\*

\* @brief AT24C02读取一个字节

\* @param WordAddress 要读出字节的地址

\* @retval 读出的数据

\*/

unsigned char AT24C02\_ReadByte(unsigned char WordAddress)

{

unsigned char Data;

I2C\_Start();

I2C\_SendByte(AT24C02\_ADDRESS);

I2C\_ReceiveAck();

I2C\_SendByte(WordAddress);

I2C\_ReceiveAck();

I2C\_Start();

I2C\_SendByte(AT24C02\_ADDRESS|0x01);

I2C\_ReceiveAck();

Data=I2C\_ReceiveByte();

I2C\_SendAck(1);

I2C\_Stop();

return Data;

}

三、DHT11温湿度传感器

**DHT11.c**

#include<reg52.h>

#include "delay.h"

#include "lcd12864.h"

#define uchar unsigned char

#define uint unsigned int

#define Data\_0\_time 4

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

//----------------IO口定义区--------------------//

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

sbit P3\_7 = P3 ^ 7;

sbit LED = P2 ^ 0;

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

//----------------定义区--------------------//

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

u8 U8FLAG, k;

u8 U8count, U8temp;

u16 U8T\_data\_H, U8T\_data\_L, U8RH\_data\_H, U8RH\_data\_L, U8checkdata;

u16 U8T\_data\_H\_temp, U8T\_data\_L\_temp, U8RH\_data\_H\_temp, U8RH\_data\_L\_temp, U8checkdata\_temp;

u8 U8comdata;

u8 outdata[5]; //定义发送的字节数

u8 indata[5];

u8 count, count\_r = 0;

u8 str[5] = { "RS232" };

u16 U16temp1, U16temp2;

u16 m, n, x, y;

u16 table[10] = { 0,1,2,3,4,5,6,7,8,9 };

void COM(void)

{

u8 i;

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

{

U8FLAG = 2;

while ((!P3\_7) && U8FLAG++);

Delay\_10us();

//Delay\_10us();

delay\_us(2);

//Delay\_10us();

U8temp = 0;

if (P3\_7) U8temp = 1;

U8FLAG = 2;

while ((P3\_7) && U8FLAG++);

//超时则跳出for循环

if (U8FLAG == 1)break;

//判断数据位是0还是1

// 如果高电平高过预定0高电平值则数据位为 1

U8comdata <<= 1;

U8comdata |= U8temp;

}

}

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

//-----湿度读取子程序 ------------

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

//----以下变量均为全局变量--------

//----温度高8位== U8T\_data\_H------

//----温度低8位== U8T\_data\_L------

//----湿度高8位== U8RH\_data\_H-----

//----湿度低8位== U8RH\_data\_L-----

//----校验 8位 == U8checkdata-----

//----调用相关子程序如下----------

//---- Delay();, Delay\_10us();,COM();

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

void RH(void)

{

//主机拉低18ms

P3\_7 = 0;

delay\_ms(5);

P3\_7 = 1;

//总线由上拉电阻拉高 主机延时20us

delay\_us(4);

//主机设为输入 判断从机响应信号

P3\_7 = 1;

//判断从机是否有低电平响应信号 如不响应则跳出，响应则向下运行

delay\_us(4);

if (!P3\_7)

{

U8FLAG = 2;

//判断从机是否发出 80us 的低电平响应信号是否结束

while ((!P3\_7) && U8FLAG++);

U8FLAG = 2;

//判断从机是否发出 80us 的高电平，如发出则进入数据接收状态

while ((P3\_7) && U8FLAG++);

//数据接收状态

COM();

U8RH\_data\_H\_temp = U8comdata;

COM();

U8RH\_data\_L\_temp = U8comdata;

COM();

U8T\_data\_H\_temp = U8comdata;

COM();

U8T\_data\_L\_temp = U8comdata;

COM();

U8checkdata\_temp = U8comdata;

P3\_7 = 1;

//数据校验

U8temp = (U8T\_data\_H\_temp + U8T\_data\_L\_temp + U8RH\_data\_H\_temp + U8RH\_data\_L\_temp);

if (U8temp == U8checkdata\_temp)

{

U8RH\_data\_H = U8RH\_data\_H\_temp;

U8RH\_data\_L = U8RH\_data\_L\_temp;

U8T\_data\_H = U8T\_data\_H\_temp;

U8T\_data\_L = U8T\_data\_L\_temp;

U8checkdata = U8checkdata\_temp;

}

}

}

u16 data\_transfer(u8 dat)

{

u16 x, y, n;

x = dat & 0x0f;

y = (dat & 0xf0) >> 4;

n = y \* 16 + x;

return n;

}

void main()//主函数模块//

{

lcd12864\_init();

delay\_ms(1500);

lcd12864\_show\_string1(0, 0, "温度");

lcd12864\_show\_string1(0, 1, "湿度");

while (1)

{

//delay\_ms(1500);

RH();

delay\_ms(1);

m = data\_transfer(U8T\_data\_H);

n = data\_transfer(U8RH\_data\_H);

x = data\_transfer(U8RH\_data\_L);

y = data\_transfer(U8T\_data\_L);

lcd12864\_show\_string2(2, 0, table[m / 10]);

lcd12864\_show\_string2(3, 0, table[m % 10]);

lcd12864\_show\_string1(4, 0, ".");

lcd12864\_show\_string2(5, 0, table[y % 10]);

lcd12864\_show\_string1(6, 0, "℃");

lcd12864\_show\_string2(2, 1, table[n / 10]);

lcd12864\_show\_string2(3, 1, table[n % 10]);

lcd12864\_show\_string1(4, 1, ".");

lcd12864\_show\_string2(5, 1, table[x % 10]);

lcd12864\_show\_string1(6, 1, "%");

}

}

四、DS18B20温度传感器（单总线）

**OneWire.h**

#ifndef \_\_ONEWIRE\_H\_\_

#define \_\_ONEWIRE\_H\_\_

unsigned char OneWire\_Init(void);

void OneWire\_SendBit(unsigned char Bit);

unsigned char OneWire\_ReceiveBit(void);

void OneWire\_SendByte(unsigned char Byte);

unsigned char OneWire\_ReceiveByte(void);

#endif

**OneWire.c**

#include <REGX52.H>

//引脚定义

sbit OneWire\_DQ=P3^7;

/\*\*

\* @brief 单总线初始化

\* @param 无

\* @retval 从机响应位，0为响应，1为未响应

\*/

unsigned char OneWire\_Init(void)

{

unsigned char i;

unsigned char AckBit;

OneWire\_DQ=1;

OneWire\_DQ=0;

i = 247;while (--i); //Delay 500us

OneWire\_DQ=1;

i = 32;while (--i); //Delay 70us

AckBit=OneWire\_DQ;

i = 247;while (--i); //Delay 500us

return AckBit;

}

/\*\*

\* @brief 单总线发送一位

\* @param Bit 要发送的位

\* @retval 无

\*/

void OneWire\_SendBit(unsigned char Bit)

{

unsigned char i;

OneWire\_DQ=0;

i = 4;while (--i); //Delay 10us

OneWire\_DQ=Bit;

i = 24;while (--i); //Delay 50us

OneWire\_DQ=1;

}

/\*\*

\* @brief 单总线接收一位

\* @param 无

\* @retval 读取的位

\*/

unsigned char OneWire\_ReceiveBit(void)

{

unsigned char i;

unsigned char Bit;

OneWire\_DQ=0;

i = 2;while (--i); //Delay 5us

OneWire\_DQ=1;

i = 2;while (--i); //Delay 5us

Bit=OneWire\_DQ;

i = 24;while (--i); //Delay 50us

return Bit;

}

/\*\*

\* @brief 单总线发送一个字节

\* @param Byte 要发送的字节

\* @retval 无

\*/

void OneWire\_SendByte(unsigned char Byte)

{

unsigned char i;

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

{

OneWire\_SendBit(Byte&(0x01<<i));

}

}

/\*\*

\* @brief 单总线接收一个字节

\* @param 无

\* @retval 接收的一个字节

\*/

unsigned char OneWire\_ReceiveByte(void)

{

unsigned char i;

unsigned char Byte=0x00;

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

{

if(OneWire\_ReceiveBit()){Byte|=(0x01<<i);}

}

return Byte;

}

**DS18B20.h**

#ifndef \_\_DS18B20\_H\_\_

#define \_\_DS18B20\_H\_\_

void DS18B20\_ConvertT(void);

float DS18B20\_ReadT(void);

#endif

**DS18B20.c**

#include <REGX52.H>

#include "OneWire.h"

//DS18B20指令

#define DS18B20\_SKIP\_ROM 0xCC

#define DS18B20\_CONVERT\_T 0x44

#define DS18B20\_READ\_SCRATCHPAD 0xBE

/\*\*

\* @brief DS18B20开始温度变换

\* @param 无

\* @retval 无

\*/

void DS18B20\_ConvertT(void)

{

OneWire\_Init();

OneWire\_SendByte(DS18B20\_SKIP\_ROM);

OneWire\_SendByte(DS18B20\_CONVERT\_T);

}

/\*\*

\* @brief DS18B20读取温度

\* @param 无

\* @retval 温度数值

\*/

float DS18B20\_ReadT(void)

{

unsigned char TLSB,TMSB;

int Temp;

float T;

OneWire\_Init();

OneWire\_SendByte(DS18B20\_SKIP\_ROM);

OneWire\_SendByte(DS18B20\_READ\_SCRATCHPAD);

TLSB=OneWire\_ReceiveByte();

TMSB=OneWire\_ReceiveByte();

Temp=(TMSB<<8)|TLSB;

T=Temp/16.0;

return T;

}

五、DS1302时钟芯片

**DS302.h**

#ifndef \_\_DS1302\_H\_\_

#define \_\_DS1302\_H\_\_

//外部可调用时间数组，索引0~6分别为年、月、日、时、分、秒、星期

extern unsigned char DS1302\_Time[];

void DS1302\_Init(void);

void DS1302\_WriteByte(unsigned char Command,Data);

unsigned char DS1302\_ReadByte(unsigned char Command);

void DS1302\_SetTime(void);

void DS1302\_ReadTime(void);

#endif

**DS1302.c**

#include <REGX52.H>

//引脚定义

sbit DS1302\_SCLK=P3^6;

sbit DS1302\_IO=P3^4;

sbit DS1302\_CE=P3^5;

//寄存器写入地址/指令定义

#define DS1302\_SECOND 0x80

#define DS1302\_MINUTE 0x82

#define DS1302\_HOUR 0x84

#define DS1302\_DATE 0x86

#define DS1302\_MONTH 0x88

#define DS1302\_DAY 0x8A

#define DS1302\_YEAR 0x8C

#define DS1302\_WP 0x8E

//时间数组，索引0~6分别为年、月、日、时、分、秒、星期

unsigned char DS1302\_Time[]={19,11,16,12,59,55,6};

/\*\*

\* @brief DS1302初始化

\* @param 无

\* @retval 无

\*/

void DS1302\_Init(void)

{

DS1302\_CE=0;

DS1302\_SCLK=0;

}

/\*\*

\* @brief DS1302写一个字节

\* @param Command 命令字/地址

\* @param Data 要写入的数据

\* @retval 无

\*/

void DS1302\_WriteByte(unsigned char Command,Data)

{

unsigned char i;

DS1302\_CE=1;

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

{

DS1302\_IO=Command&(0x01<<i);

DS1302\_SCLK=1;

DS1302\_SCLK=0;

}

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

{

DS1302\_IO=Data&(0x01<<i);

DS1302\_SCLK=1;

DS1302\_SCLK=0;

}

DS1302\_CE=0;

}

/\*\*

\* @brief DS1302读一个字节

\* @param Command 命令字/地址

\* @retval 读出的数据

\*/

unsigned char DS1302\_ReadByte(unsigned char Command)

{

unsigned char i,Data=0x00;

Command|=0x01; //将指令转换为读指令

DS1302\_CE=1;

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

{

DS1302\_IO=Command&(0x01<<i);

DS1302\_SCLK=0;

DS1302\_SCLK=1;

}

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

{

DS1302\_SCLK=1;

DS1302\_SCLK=0;

if(DS1302\_IO){Data|=(0x01<<i);}

}

DS1302\_CE=0;

DS1302\_IO=0; //读取后将IO设置为0，否则读出的数据会出错

return Data;

}

/\*\*

\* @brief DS1302设置时间，调用之后，DS1302\_Time数组的数字会被设置到DS1302中

\* @param 无

\* @retval 无

\*/

void DS1302\_SetTime(void)

{

DS1302\_WriteByte(DS1302\_WP,0x00);

DS1302\_WriteByte(DS1302\_YEAR,DS1302\_Time[0]/10\*16+DS1302\_Time[0]%10);//十进制转BCD码后写入

DS1302\_WriteByte(DS1302\_MONTH,DS1302\_Time[1]/10\*16+DS1302\_Time[1]%10);

DS1302\_WriteByte(DS1302\_DATE,DS1302\_Time[2]/10\*16+DS1302\_Time[2]%10);

DS1302\_WriteByte(DS1302\_HOUR,DS1302\_Time[3]/10\*16+DS1302\_Time[3]%10);

DS1302\_WriteByte(DS1302\_MINUTE,DS1302\_Time[4]/10\*16+DS1302\_Time[4]%10);

DS1302\_WriteByte(DS1302\_SECOND,DS1302\_Time[5]/10\*16+DS1302\_Time[5]%10);

DS1302\_WriteByte(DS1302\_DAY,DS1302\_Time[6]/10\*16+DS1302\_Time[6]%10);

DS1302\_WriteByte(DS1302\_WP,0x80);

}

/\*\*

\* @brief DS1302读取时间，调用之后，DS1302中的数据会被读取到DS1302\_Time数组中

\* @param 无

\* @retval 无

\*/

void DS1302\_ReadTime(void)

{

unsigned char Temp;

Temp=DS1302\_ReadByte(DS1302\_YEAR);

DS1302\_Time[0]=Temp/16\*10+Temp%16;//BCD码转十进制后读取

Temp=DS1302\_ReadByte(DS1302\_MONTH);

DS1302\_Time[1]=Temp/16\*10+Temp%16;

Temp=DS1302\_ReadByte(DS1302\_DATE);

DS1302\_Time[2]=Temp/16\*10+Temp%16;

Temp=DS1302\_ReadByte(DS1302\_HOUR);

DS1302\_Time[3]=Temp/16\*10+Temp%16;

Temp=DS1302\_ReadByte(DS1302\_MINUTE);

DS1302\_Time[4]=Temp/16\*10+Temp%16;

Temp=DS1302\_ReadByte(DS1302\_SECOND);

DS1302\_Time[5]=Temp/16\*10+Temp%16;

Temp=DS1302\_ReadByte(DS1302\_DAY);

DS1302\_Time[6]=Temp/16\*10+Temp%16;

}

六、GY302光强传感器（I2C通信）

**BH1750.h**

#include <REGX52.H>

#include "Delay.h"

#include "LCD1602.h"

#include <math.h> //Keil library

#include <stdio.h> //Keil library

#define uchar unsigned char

#define uint unsigned int

sbit SCL = P1 ^ 0; //IIC时钟引脚定义

sbit SDA = P1 ^ 1; //IIC数据引脚定义

//VCC--3.3V

//GND--GND

//ADDR不接

#define SlaveAddress 0x46 //定义器件在IIC总线中的从地址,根据ALT ADDRESS地址引脚不同修改

//ALT ADDRESS引脚接地时地址为0x46，接电源时地址为0xB8

uchar BUF[8]; //接收数据缓存区

uchar ge, shi, bai, qian, wan; //显示变量

int dis\_data; //变量

void BH1750\_Init(void);

void conversion(uint temp\_data);

void Single\_Write\_BH1750(uchar REG\_Address); //单个写入数据

uchar Single\_Read\_BH1750(uchar REG\_Address); //单个读取内部寄存器数据

void Multiple\_Read\_BH1750(); //连续的读取内部寄存器数据

void BH1750\_Start(); //起始信号

void BH1750\_Stop(); //停止信号

void BH1750\_SendACK(bit ack); //应答ACK

bit BH1750\_RecvACK(); //读ack

void BH1750\_SendByte(unsigned char dat); //IIC单个字节写

unsigned char BH1750\_RecvByte(); //IIC单个字节读

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

void conversion(uint temp\_data) // 数据转换出 个，十，百，千，万

{

wan = temp\_data / 10000 + 0x30;

temp\_data = temp\_data % 10000; //取余运算

qian = temp\_data / 1000 + 0x30;

temp\_data = temp\_data % 1000; //取余运算

bai = temp\_data / 100 + 0x30;

temp\_data = temp\_data % 100; //取余运算

shi = temp\_data / 10 + 0x30;

temp\_data = temp\_data % 10; //取余运算

ge = temp\_data + 0x30;

}

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

起始信号

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

void BH1750\_Start()

{

SDA = 1; //拉高数据线

SCL = 1; //拉高时钟线

SDA = 0; //产生下降沿

SCL = 0; //拉低时钟线

}

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

停止信号

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

void BH1750\_Stop()

{

SDA = 0; //拉低数据线

SCL = 1; //拉高时钟线

SDA = 1; //产生上升沿

}

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

发送应答信号

入口参数:ack (0:ACK 1:NAK)

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

void BH1750\_SendACK(bit ack)

{

SDA = ack; //写应答信号

SCL = 1; //拉高时钟线

SCL = 0; //拉低时钟线

}

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

接收应答信号

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

bit BH1750\_RecvACK()

{

SCL = 1; //拉高时钟线

CY = SDA; //读应答信号

SCL = 0; //拉低时钟线

return CY;

}

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

向IIC总线发送一个字节数据

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

void BH1750\_SendByte(uchar dat)

{

unsigned char i;

for (i = 0; i < 8; i++) //8位计数器

{

dat <<= 1; //移出数据的最高位

SDA = CY; //送数据口

SCL = 1; //拉高时钟线

SCL = 0; //拉低时钟线

}

BH1750\_RecvACK();

}

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

从IIC总线接收一个字节数据

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

unsigned char BH1750\_RecvByte()

{

uchar i;

uchar dat = 0;

SDA = 1; //使能内部上拉,准备读取数据,

for (i = 0; i < 8; i++) //8位计数器

{

dat <<= 1;

SCL = 1; //拉高时钟线

dat |= SDA; //读数据

SCL = 0; //拉低时钟线

}

return dat;

}

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

void Single\_Write\_BH1750(uchar REG\_Address)

{

BH1750\_Start(); //起始信号

BH1750\_SendByte(SlaveAddress); //发送设备地址+写信号

BH1750\_SendByte(REG\_Address); //内部寄存器地址，

// BH1750\_SendByte(REG\_data); //内部寄存器数据，

BH1750\_Stop(); //发送停止信号

}

//\*\*\*\*\*\*\*\*单字节读取\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*

uchar Single\_Read\_BH1750(uchar REG\_Address)

{ uchar REG\_data;

BH1750\_Start(); //起始信号

BH1750\_SendByte(SlaveAddress); //发送设备地址+写信号

BH1750\_SendByte(REG\_Address); //发送存储单元地址，从0开始

BH1750\_Start(); //起始信号

BH1750\_SendByte(SlaveAddress+1); //发送设备地址+读信号

REG\_data=BH1750\_RecvByte(); //读出寄存器数据

BH1750\_SendACK(1);

BH1750\_Stop(); //停止信号

return REG\_data;

}

\*/

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

//连续读出BH1750内部数据

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

void Multiple\_read\_BH1750(void)

{

uchar i;

BH1750\_Start(); //起始信号

BH1750\_SendByte(SlaveAddress + 1); //发送设备地址+读信号

for (i = 0; i < 3; i++) //连续读取2个地址数据，存储中BUF

{

BUF[i] = BH1750\_RecvByte(); //BUF[0]存储0x32地址中的数据

if (i == 3)

BH1750\_SendACK(1); //最后一个数据需要回NOACK

else

BH1750\_SendACK(0); //回应ACK

}

BH1750\_Stop(); //停止信号

Delay(5);

}

//初始化BH1750，根据需要请参考pdf进行修改

void BH1750\_Init()

{

Single\_Write\_BH1750(0x01);

}

七、I2C总线

**I2C.h**

#ifndef \_\_I2C\_H\_\_

#define \_\_I2C\_H\_\_

#include <REGX52.H>

sbit I2C\_SCL = P2 ^ 1;

sbit I2C\_SDA = P2 ^ 0;

//函数声明

void I2C\_Start(void);

void I2C\_Stop(void);

void I2C\_SendByte(unsigned char Byte);

unsigned char I2C\_ReceiveByte(void);

void I2C\_SendAck(unsigned char AckBit);

unsigned char I2C\_ReceiveAck(void);

/\*\*

\* @brief I2C开始

\* @param 无

\* @retval 无

\*/

void I2C\_Start(void)

{

I2C\_SDA = 1;

I2C\_SCL = 1;

I2C\_SDA = 0;

I2C\_SCL = 0;

}

/\*\*

\* @brief I2C停止

\* @param 无

\* @retval 无

\*/

void I2C\_Stop(void)

{

I2C\_SDA = 0;

I2C\_SCL = 1;

I2C\_SDA = 1;

}

/\*\*

\* @brief I2C发送一个字节

\* @param Byte 要发送的字节

\* @retval 无

\*/

void I2C\_SendByte(unsigned char Byte)

{

unsigned char i;

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

{

I2C\_SDA = Byte & (0x80 >> i);

I2C\_SCL = 1;

I2C\_SCL = 0;

}

}

/\*\*

\* @brief I2C接收一个字节

\* @param 无

\* @retval 接收到的一个字节数据

\*/

unsigned char I2C\_ReceiveByte(void)

{

unsigned char i, Byte = 0x00;

I2C\_SDA = 1;

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

{

I2C\_SCL = 1;

if (I2C\_SDA) { Byte |= (0x80 >> i); }

I2C\_SCL = 0;

}

return Byte;

}

/\*\*

\* @brief I2C发送应答

\* @param AckBit 应答位，0为应答，1为非应答

\* @retval 无

\*/

void I2C\_SendAck(unsigned char AckBit)

{

I2C\_SDA = AckBit;

I2C\_SCL = 1;

I2C\_SCL = 0;

}

/\*\*

\* @brief I2C接收应答位

\* @param 无

\* @retval 接收到的应答位，0为应答，1为非应答

\*/

unsigned char I2C\_ReceiveAck(void)

{

unsigned char AckBit;

I2C\_SDA = 1;

I2C\_SCL = 1;

AckBit = I2C\_SDA;

I2C\_SCL = 0;

return AckBit;

}

#endif

八、LCD12864显示屏

**lcd12864.h**

#include "main.h"

#include "delay.h"

sbit lcd12864\_RS = P2 ^ 6;//数据命令选择

sbit lcd12864\_RW = P2 ^ 5;//读写选择

sbit lcd12864\_E = P2 ^ 7;//使能信号

sbit lcd12864\_PSB = P3 ^ 2;//:8位或4位并口/串口选择

#define lcd12864\_data P0 //LCD12864数据端口定义

void lcd12864\_write\_data(u8 dat);

void lcd12864\_write\_cmd(u8 cmd);

void lcd12864\_init();

void lcd12864\_clear();

void lcd12864\_show\_string1(u8 x, u8 y, u8\* str);

void lcd12864\_show\_string2(u8 x, u8 y, u8 m);

**lcd12864.c**

#include "lcd12864.h"

//u8 lcd12864\_check\_busy() //:忙检测

//{

// lcd12864\_data = 0xff;

// lcd12864\_RS = 0;

// lcd12864\_RW = 1;

// lcd12864\_E = 1;

// delay\_ms(1);

// data1 = lcd12864\_data;

// lcd12864\_E = 0;

// return data1 & 0x80;

//}

void lcd12864\_write\_cmd(u8 cmd)

{

// while(lcd12864\_check\_busy())

// {

lcd12864\_RS = 0;//选择命令

lcd12864\_RW = 0;//选择写

lcd12864\_E = 0;

lcd12864\_data = cmd;//准备命令

delay\_ms(1);

lcd12864\_E = 1;//使能脚E先上升沿写入

delay\_ms(1);

lcd12864\_E = 0;//使能脚E后负跳变完成写入

// }

}

void lcd12864\_write\_data(u8 dat)

{

// while(lcd12864\_check\_busy())

// {

lcd12864\_RS = 1;//选择数据

lcd12864\_RW = 0;//选择写

lcd12864\_E = 0;

lcd12864\_data = dat;//准备数据

delay\_ms(1);

lcd12864\_E = 1;//使能脚E先上升沿写入

delay\_ms(1);

lcd12864\_E = 0;//使能脚E后负跳变完成写入

// }

}

void lcd12864\_init()

{

lcd12864\_PSB = 1;//选择8位或4位并口方式

lcd12864\_write\_cmd(0x30);//数据总线8位，基本指令操作

lcd12864\_write\_cmd(0x0c);//整体显示关，游标显示关，游标正常显示

lcd12864\_write\_cmd(0x06);//写入新数据后光标右移，显示屏不移动

lcd12864\_write\_cmd(0x01);//清屏

// lcd12864\_write\_cmd(0x80);//设定DDRAM位址到位址计数器AC

// lcd12864\_write\_cmd(0x34);//扩充指令集；绘图

// lcd12864\_write\_cmd(0x02);//地址归位

}

void lcd12864\_clear()//清屏

{

lcd12864\_write\_cmd(0x01);

}

void lcd12864\_show\_string1(u8 x, u8 y, u8\* str)

{

if (y <= 0)

y = 0;

if (y > 3)

y = 3;

x &= 0x0f; //限制x,y不能大于显示范围

switch (y)

{

case 0: x |= 0x80; break;//第1行地址+x的偏移

case 1: x |= 0x90; break;//第2行地址+x的偏移

case 2: x |= 0x88; break;//第3行地址+x的偏移

case 3: x |= 0x98; break;//第4行地址+x的偏移

}

lcd12864\_write\_cmd(x);

while (\*str != '\0')

{

lcd12864\_write\_data(\*str);

str++;

}

}

void lcd12864\_show\_string2(u8 x, u8 y, u8 m)

{

if (y <= 0)

y = 0;

if (y > 3)

y = 3;

x &= 0x0f; //限制x,y不能大于显示范围

switch (y)

{

case 0: x |= 0x80; break;//第1行地址+x的偏移

case 1: x |= 0x90; break;//第2行地址+x的偏移

case 2: x |= 0x88; break;//第3行地址+x的偏移

case 3: x |= 0x98; break;//第4行地址+x的偏移

}

lcd12864\_write\_cmd(x);

lcd12864\_write\_data('0' + m);

}

九、MP-4可燃气体检测

**MP-4.c(示例)**

#include<reg52.h>

#include "main.h"

#include "lcd12864.h"

#include "delay.h"

#include "ADC.h"

u8 num[4];

float s;

u16 p, temp;

void main()

{

lcd12864\_init();

while (1)

{

temp = read\_AD\_dat(0xe4);

s = 0.00122 \* temp;

p = s \* 1000;

num[0] = p / 1000;

num[1] = (p % 1000) / 100;

num[2] = ((p % 1000) % 100) / 10;

num[3] = p % 10;

lcd12864\_show\_string1(0, 0, "当前电压值为：");

lcd12864\_show\_string2(0, 1, num[0]);

lcd12864\_show\_string1(1, 1, ".");

lcd12864\_show\_string2(2, 1, num[1]);

lcd12864\_show\_string2(3, 1, num[2]);

lcd12864\_show\_string2(4, 1, num[3]);

lcd12864\_show\_string1(5, 1, "V");

}

}

十、PWM(示例)

**breath\_light.h**

#ifndef \_breath\_light\_H

#define \_breath\_light\_H

#include "main.h"

#include "led\_fuc.h"

void breath\_light1(u16 datas);

#endif

**breath\_light.c**

#include<reg52.h>

#include "breath\_light.h"

u8 PWM\_COUNT = 0; //:计数，通过该值判断已经进入了多少次中断，并与taget\_value比对

u16 update\_count = 0; //:多久更新一次PWM值，太快的更新将导致呼吸灯亮度变化不明显

u8 taget\_value = 50; //:欲达到的占空比值

u8 flag;

void breath\_light1(u16 datas)

{

PWM\_COUNT = (PWM\_COUNT + 1) % 100;

update\_count++;

if (PWM\_COUNT <= taget\_value)//Ton是否已经结束

{

switch (datas)

{

case(1):led\_fuc(1); break;

case(2):led\_fuc(2); break;

case(3):led\_fuc(3); break;

case(4):led\_fuc(4); break;

case(5):led\_fuc(5); break;

case(6):led\_fuc(6); break;

}

}

else

{

LED1 = 1;

LED2 = 1;

LED3 = 1;

LED4 = 1;

}

if ((update\_count == 50) && (flag == 0))

{

update\_count = 0;

taget\_value++;

if (taget\_value >= 99) //已经达到最亮

flag = 1; //转为变暗

}

if ((update\_count == 50) && (flag == 1))

{

update\_count = 0;

taget\_value--;

if (taget\_value <= 1) //已经达到最暗

flag = 0; //转为变亮

}

}

十一、RC522(RFID射频IC卡感应)

**MAIN.H**

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

//端口定义

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

//MFRC522

sbit MF522\_RST = P1^4; //RC500片选

sbit MF522\_NSS = P1^2;

sbit MF522\_SCK = P1^1;

sbit MF522\_SI = P1^0;

sbit MF522\_SO = P1^3;

//指示灯

sbit LED\_GREEN = P0^0;

sbit LED1 = P0^1;

sbit LED2 = P0^2;

sbit LED3 = P0^3;

sbit LED4 = P0^4;

sbit LED5 = P0^5;

sbit LED6 = P0^6;

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

//函数原型

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

void InitializeSystem();

#define BAUD\_115200 256 - (OSC\_FREQ/192L)/115200L // 255

#define BAUD\_57600 256 - (OSC\_FREQ/192L)/57600L // 254

#define BAUD\_38400 256 - (OSC\_FREQ/192L)/38400L // 253

#define BAUD\_28800 256 - (OSC\_FREQ/192L)/28800L // 252

#define BAUD\_19200 256 - (OSC\_FREQ/192L)/19200L // 250

#define BAUD\_14400 256 - (OSC\_FREQ/192L)/14400L // 248

#define BAUD\_9600 256 - (OSC\_FREQ/192L)/9600L // 244

//#define OSC\_FREQ 22118400L

//#define OSC\_FREQ 7372800L

#define OSC\_FREQ 11059200L

#define RCAP2\_50us 65536L - OSC\_FREQ/40417L

#define RCAP2\_1ms 65536L - OSC\_FREQ/2000L

#define RCAP2\_10ms 65536L - OSC\_FREQ/1200L

#define TIME0\_500us 65536L - OSC\_FREQ/8000L

#define TIME0\_10ms 65536L - OSC\_FREQ/200

#define CALL\_isr\_UART() TI = 1

#define TRUE 1

#define FALSE 0

**MAIN.C**

#include "reg52.h"

#include "main.h"

#include "LCD1602.h"

#include "Timer0.h"

#include "mfrc522.h"

#include <string.h>

//M1卡的某一块写为如下格式，则该块为钱包，可接收扣款和充值命令

//4字节金额（低字节在前）＋4字节金额取反＋4字节金额＋1字节块地址＋1字节块地址取反＋1字节块地址＋1字节块地址取反

unsigned char code data1[16] = { 0x12,0x34,0x56,0x78,0xED,0xCB,0xA9,0x87,0x12,0x34,0x56,0x78,0x01,0xFE,0x01,0xFE };

unsigned char code data2[4] = { 0x12,0,0,0 };

unsigned char code DefaultKey[6] = { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF }; //默认密码

unsigned char g\_ucTempbuf[20];

unsigned char idata MLastSelectedSnr[4];

unsigned char idata RevBuffer[30]; //寄存器指令缓存

unsigned char data SerBuffer[20]; //数据缓存

bit CmdValid = 1;

void main()

{

unsigned char i = 0;

unsigned char status;

InitializeSystem();

Timer0\_Init();

LCD\_Init();

LCD\_ShowString(1, 1, "ID:");

while (1)

{

status = PcdRequest(PICC\_REQALL, g\_ucTempbuf);//寻卡

//加入定时器，当检测到卡时，不计时

if(status == MI\_OK)

{

TR0 = 0;

T0Count = 0;

}

//未检测到卡5s后，第二行清屏

if(status != MI\_OK)

{

TR0 = 1;

if(T0Count>=30000)

{

LCD\_ShowString(2, 1, " ");

TR0 = 0;

T0Count = 0;

}

}

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

if (status != MI\_OK)

{

LED\_GREEN = 1;

continue;

}

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

LED\_GREEN = 0;

status = PcdAnticoll(g\_ucTempbuf);//防冲撞

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

if (status != MI\_OK)

{

continue;

}

///////////////////////////打印序列号///////////////////////////////////

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

{

LCD\_ShowHexNum(2, 2\*i+1, g\_ucTempbuf[i],2);

}

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

status = PcdSelect(g\_ucTempbuf);//选定读卡区内某一张卡//参数就是序列号

if (status != MI\_OK)

{

continue;

}

LED3 = 0;

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

status = PcdAuthState(PICC\_AUTHENT1A, 1, DefaultKey, g\_ucTempbuf);//验证密码

if (status != MI\_OK)

{

continue;

}

LED4 = 0;

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

status = PcdWrite(1, data1);//

if (status != MI\_OK)

{

continue;

}

LED5 = 0;

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

status = PcdValue(PICC\_DECREMENT, 1, data2);//扣费

if (status != MI\_OK)

{

continue;

}

LED6 = 0;

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

status = PcdBakValue(1, 2);//备份钱包

if (status != MI\_OK)

{

continue;

}

LED1 = 0;

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

status = PcdRead(2, g\_ucTempbuf);//读值，目测是校验扣费是否成功

if (status != MI\_OK)

{

continue;

}

LED2 = 0;

PcdHalt();//命令卡片进入休眠状态

}

}

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

//系统初始化

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

void InitializeSystem()

{

P0 = 0xFF; P1 = 0xFF; P3 = 0xFF;

CmdValid = 1;

LED\_GREEN = 1;

PcdReset();

PcdAntennaOff();

PcdAntennaOn();

M500PcdConfigISOType('A');

LED\_GREEN = 0;

delay\_10ms(10);

LED\_GREEN = 1;

delay\_10ms(10);

LED\_GREEN = 0;

}

**MFRC522.H**

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

//函数原型

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

char PcdReset(void);

void PcdAntennaOn(void);

void PcdAntennaOff(void);

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

char PcdAnticoll(unsigned char \*pSnr);

char PcdSelect(unsigned char \*pSnr);

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

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

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

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

char PcdBakValue(unsigned char sourceaddr, unsigned char goaladdr);

char PcdHalt(void);

char PcdComMF522(unsigned char Command,

unsigned char \*pInData,

unsigned char InLenByte,

unsigned char \*pOutData,

unsigned int \*pOutLenBit);

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

void WriteRawRC(unsigned char Address,unsigned char value);

unsigned char ReadRawRC(unsigned char Address);

void SetBitMask(unsigned char reg,unsigned char mask);

void ClearBitMask(unsigned char reg,unsigned char mask);

char M500PcdConfigISOType(unsigned char type);

void delay\_10ms(unsigned int \_10ms);

void iccardcode();

char PcdBakValue(unsigned char sourceaddr, unsigned char goaladdr);

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

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

//MF522命令字

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

#define PCD\_IDLE 0x00 //取消当前命令

#define PCD\_AUTHENT 0x0E //验证密钥

#define PCD\_RECEIVE 0x08 //接收数据

#define PCD\_TRANSMIT 0x04 //发送数据

#define PCD\_TRANSCEIVE 0x0C //发送并接收数据

#define PCD\_RESETPHASE 0x0F //复位

#define PCD\_CALCCRC 0x03 //CRC计算

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

//Mifare\_One卡片命令字

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

#define PICC\_REQIDL 0x26 //寻天线区内未进入休眠状态

#define PICC\_REQALL 0x52 //寻天线区内全部卡

#define PICC\_ANTICOLL1 0x93 //防冲撞

#define PICC\_ANTICOLL2 0x95 //防冲撞

#define PICC\_AUTHENT1A 0x60 //验证A密钥

#define PICC\_AUTHENT1B 0x61 //验证B密钥

#define PICC\_READ 0x30 //读块

#define PICC\_WRITE 0xA0 //写块

#define PICC\_DECREMENT 0xC0 //扣款

#define PICC\_INCREMENT 0xC1 //充值

#define PICC\_RESTORE 0xC2 //调块数据到缓冲区

#define PICC\_TRANSFER 0xB0 //保存缓冲区中数据

#define PICC\_HALT 0x50 //休眠

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

//MF522 FIFO长度定义

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

#define DEF\_FIFO\_LENGTH 64 //FIFO size=64byte

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

//MF522寄存器定义

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

// PAGE 0

#define RFU00 0x00

#define CommandReg 0x01

#define ComIEnReg 0x02

#define DivlEnReg 0x03

#define ComIrqReg 0x04

#define DivIrqReg 0x05

#define ErrorReg 0x06

#define Status1Reg 0x07

#define Status2Reg 0x08

#define FIFODataReg 0x09

#define FIFOLevelReg 0x0A

#define WaterLevelReg 0x0B

#define ControlReg 0x0C

#define BitFramingReg 0x0D

#define CollReg 0x0E

#define RFU0F 0x0F

// PAGE 1

#define RFU10 0x10

#define ModeReg 0x11

#define TxModeReg 0x12

#define RxModeReg 0x13

#define TxControlReg 0x14

#define TxAutoReg 0x15

#define TxSelReg 0x16

#define RxSelReg 0x17

#define RxThresholdReg 0x18

#define DemodReg 0x19

#define RFU1A 0x1A

#define RFU1B 0x1B

#define MifareReg 0x1C

#define RFU1D 0x1D

#define RFU1E 0x1E

#define SerialSpeedReg 0x1F

// PAGE 2

#define RFU20 0x20

#define CRCResultRegM 0x21

#define CRCResultRegL 0x22

#define RFU23 0x23

#define ModWidthReg 0x24

#define RFU25 0x25

#define RFCfgReg 0x26

#define GsNReg 0x27

#define CWGsCfgReg 0x28

#define ModGsCfgReg 0x29

#define TModeReg 0x2A

#define TPrescalerReg 0x2B

#define TReloadRegH 0x2C

#define TReloadRegL 0x2D

#define TCounterValueRegH 0x2E

#define TCounterValueRegL 0x2F

// PAGE 3

#define RFU30 0x30

#define TestSel1Reg 0x31

#define TestSel2Reg 0x32

#define TestPinEnReg 0x33

#define TestPinValueReg 0x34

#define TestBusReg 0x35

#define AutoTestReg 0x36

#define VersionReg 0x37

#define AnalogTestReg 0x38

#define TestDAC1Reg 0x39

#define TestDAC2Reg 0x3A

#define TestADCReg 0x3B

#define RFU3C 0x3C

#define RFU3D 0x3D

#define RFU3E 0x3E

#define RFU3F 0x3F

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

//和MF522通讯时返回的错误代码

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

#define MI\_OK 0

#define MI\_NOTAGERR (-1)

#define MI\_ERR (-2)

**RC522.c**

#include <intrins.h>

#include "reg52.h"

#include "main.h"

#include "mfrc522.h"

#include <string.h>

#define MAXRLEN 18

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

//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 i, ucComMF522Buf[MAXRLEN];

ucComMF522Buf[0] = auth\_mode;

ucComMF522Buf[1] = addr;

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

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

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

ucComMF522Buf[i + 8] = \*(pSnr + i);

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 i, 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))

{

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

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

}

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 i, 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);

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

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

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;

}

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

//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：复位RC522

//返 回: 成功返回MI\_OK

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

char PcdReset(void)

{

MF522\_RST = 1;

\_nop\_();

MF522\_RST = 0;

\_nop\_();

MF522\_RST = 1;

\_nop\_();

WriteRawRC(CommandReg, PCD\_RESETPHASE);

\_nop\_();

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);

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\_10ms(1);

PcdAntennaOn();

}

else { return -1; }

return MI\_OK;

}

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

//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 i, ucAddr;

unsigned char ucResult = 0;

MF522\_SCK = 0;

MF522\_NSS = 0;

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

for (i = 8; i > 0; i--)

{

MF522\_SI = ((ucAddr & 0x80) == 0x80);

MF522\_SCK = 1;

ucAddr <<= 1;

MF522\_SCK = 0;

}

for (i = 8; i > 0; i--)

{

MF522\_SCK = 1;

ucResult <<= 1;

ucResult |= (bit)MF522\_SO;

MF522\_SCK = 0;

}

MF522\_NSS = 1;

MF522\_SCK = 1;

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 i, ucAddr;

MF522\_SCK = 0;

MF522\_NSS = 0;

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

for (i = 8; i > 0; i--)

{

MF522\_SI = ((ucAddr & 0x80) == 0x80);

MF522\_SCK = 1;

ucAddr <<= 1;

MF522\_SCK = 0;

}

for (i = 8; i > 0; i--)

{

MF522\_SI = ((value & 0x80) == 0x80);

MF522\_SCK = 1;

value <<= 1;

MF522\_SCK = 0;

}

MF522\_NSS = 1;

MF522\_SCK = 1;

}

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

//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 = 0x0;

tmp = ReadRawRC(reg);

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

}

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

//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 = 0x0;

tmp = ReadRawRC(reg);

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

}

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

//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;

}

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

//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()

{

unsigned char i;

i = ReadRawRC(TxControlReg);

if (!(i & 0x03))

SetBitMask(TxControlReg, 0x03);

}

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

//关闭天线

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

void PcdAntennaOff()

{

ClearBitMask(TxControlReg, 0x03);

}

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

//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];

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);

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;

}

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

// Delay 10ms

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

void delay\_10ms(unsigned int \_10ms)

{

while (\_10ms--)

{

unsigned char i, j;

i = 18;

j = 235;

do

{

while (--j);

} while (--i);

}

}

十二、SYN6288语音合成

**SYN6288.h**

#ifndef \_\_SYN6288\_H\_\_

#define \_\_SYN6288\_H\_\_

#include<reg52.h>

void SYN\_FrameInfo(unsigned char Music, unsigned char \*HZdata);

void YS\_SYN\_Set(unsigned char \*Command);

#endif

**SYN6288.c**

#include <reg52.h>

#include <string.h>

#include "uart.h"

#include "syn6288.h"

/\*

/芯片设置命令/

unsigned char SYN\_StopCom[] = {0xFD, 0X00, 0X02, 0X02, 0XFD}; //停止合成

unsigned char SYN\_SuspendCom[] = {0XFD, 0X00, 0X02, 0X03, 0XFC}; //暂停合成

unsigned char SYN\_RecoverCom[] = {0XFD, 0X00, 0X02, 0X04, 0XFB}; //恢复合成

unsigned char SYN\_ChackCom[] = {0XFD, 0X00, 0X02, 0X21, 0XDE}; //状态查询

unsigned char SYN\_PowerDownCom[] = {0XFD, 0X00, 0X02, 0X88, 0X77}; //进入POWER DOWN 状态命令

\*/

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

//功能：合成语音 //

//参数：Music: 0:无背景音乐 1~15:选择背景音乐 //

// \*HZdata: 要合成的文字 //

//示例：SYN\_FrameInfo(0, "[v3][t3]大家好"); //

//说明： //

// 选择背景音乐2。(0：无背景音乐 1-15：背景音乐可选) //

// m[0~16]:0背景音乐为静音，16背景音乐音量最大 //

// v[0~16]:0朗读音量为静音，16朗读音量最大 //

// t[0~ 5]:0朗读语速最慢，5朗读语速最快 //

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

void SYN\_FrameInfo(unsigned char Music, unsigned char\* HZdata)

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*需要发送的文本\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned char Frame\_Info[50];

unsigned char HZ\_Length;

unsigned char ecc = 0; //定义校验字节

unsigned int i = 0;

HZ\_Length = strlen(HZdata); //需要发送文本的长度

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*帧固定配置信息\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Frame\_Info[0] = 0xFD; //构造帧头FD

Frame\_Info[1] = 0x00; //构造数据区长度的高字节

Frame\_Info[2] = HZ\_Length + 3; //构造数据区长度的低字节

Frame\_Info[3] = 0x01; //构造命令字：合成播放命令

Frame\_Info[4] = 0x01 | Music << 4; //构造命令参数：编码方式为GBK，以及背景音乐

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*校验码计算\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

for (i = 0; i < 5; i++) //依次发送构造好的5个帧头字节

{

ecc ^= Frame\_Info[i]; //对发送的字节进行异或校验

}

for (i = 0; i < HZ\_Length; i++) //依次发送待合成的文本数据

{

ecc ^= HZdata[i]; //对发送的字节进行异或校验

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*发送帧信息\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

memcpy(&Frame\_Info[5], HZdata, HZ\_Length);

Frame\_Info[HZ\_Length + 5] = ecc;

UART1\_SendString(Frame\_Info, HZ\_Length + 6);

}

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

//功能： //

// 1.停止合成：SYN\_StopCom //

// 2.暂停合成：SYN\_SuspendCom //

// 3.恢复合成：SYN\_RecoverCom //

// 4.状态查询：SYN\_ChackCom //

// 5.进入POWER DOWN状态命令：SYN\_PowerDownCom //

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

void YS\_SYN\_Set(unsigned char\* Command)

{

UART1\_SendString(Command, 6);

}

**SYN6288\_voice.c(示例)**

#include <reg52.h>

#include "syn6288.h"

#include "delay.h"

#include "uart.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*芯片设置命令\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned char SYN\_StopCom[] = {0xFD, 0X00, 0X02, 0X02, 0XFD}; //停止合成

unsigned char SYN\_SuspendCom[] = {0XFD, 0X00, 0X02, 0X03, 0XFC}; //暂停合成

unsigned char SYN\_RecoverCom[] = {0XFD, 0X00, 0X02, 0X04, 0XFB}; //恢复合成

unsigned char SYN\_ChackCom[] = {0XFD, 0X00, 0X02, 0X21, 0XDE}; //状态查询

unsigned char SYN\_PowerDownCom[] = {0XFD, 0X00, 0X02, 0X88, 0X77}; //进入POWER DOWN状态命令

unsigned char receive\_ok = 0; //接收标志位

unsigned char command; //接收串口来的数据

void main(void)

{

UART1\_Init();

SYN\_FrameInfo(0, "[v3][t3]大家好"); //喇叭说: 大家好

delay(2000);

receive\_ok = 1;

while(1)

{

/\*判断指令\*/

switch(command)

{

case '1': //如果是 1

receive\_ok = 0; //禁止接收串口数据

SYN\_FrameInfo(0, "[v3][t3]你好"); //喇叭说 你好

delay(2000);

command = 0;

receive\_ok = 1; //允许接收串口数据

break;

case 'G': //如果你 G 喇叭说: 这个是干垃圾

receive\_ok = 0; //禁止接收串口数据

SYN\_FrameInfo(0, "[v3][t3]这个是干垃圾");

delay(2000);

command = 0;

receive\_ok = 1; //允许接收串口数据

break;

}

YS\_SYN\_Set(SYN\_SuspendCom);//暂停合成

}

}

//串口1中断

void Uart1\_Isr() interrupt 4

{

if(RI && receive\_ok) //如果可以接收数据了

{

command = SBUF; //接收数据

while(!RI);

RI=0;

receive\_ok = 0;

}

}

十三、步进电机

**step\_motor.c**

/\*接线方式:

IN1 ---- P10

IN2 ---- P11

IN3 ---- P12

IN4 ---- P13

+ ---- +5V

- ---- GND

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

#include<reg52.h>

#include "delay.h"

#include "main.h"

#define MotorData P1 //步进电机控制接口定义

u8 phasecw[4] = { 0x08,0x04,0x02,0x01 }; //正转 电机导通相序 D-C-B-A

u8 phaseccw[4] = { 0x01,0x02,0x04,0x08 }; //反转 电机导通相序 A-B-C-D

u8 speed;

//speed控制电机速度

void MotorCW(void)

{

u8 i;

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

{

MotorData = phasecw[i];

delay\_ms(speed);//转速调节

}

}

//停止转动

void MotorStop(void)

{

MotorData = 0x00;

}

//主函数

void main(void)

{

u16 i;

delay\_ms(100);//等待系统稳定

speed = 50;

while (1)

{

for (i = 0; i < 10; i++)

{

MotorCW(); //顺时针转动

}

speed--; //减速

if (speed < 4)

{

speed = 50; //重新开始减速运动

MotorStop();

delay\_ms(500);

}

}

}

十四、蜂鸣器

注意：普中A2型号的蜂鸣器引脚为P2^5

**Buzzer.h**

#ifndef \_\_BUZZER\_H\_\_

#define \_\_BUZZER\_H\_\_

void Buzzer\_Time(unsigned int ms);

#endif

**Buzzer.c**

#include <REGX52.H>

#include <INTRINS.H>

//蜂鸣器端口：

sbit Buzzer = P1 ^ 5;

/\*\*

\* @brief 蜂鸣器私有延时函数，延时500us

\* @param 无

\* @retval 无

\*/

void Buzzer\_Delay500us() //@12.000MHz

{

unsigned char i;

\_nop\_();

i = 247;

while (--i);

}

/\*\*

\* @brief 蜂鸣器发声

\* @param ms 发声的时长，范围：0~32767

\* @retval 无

\*/

void Buzzer\_Time(unsigned int ms)

{

unsigned int i;

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

{

Buzzer = !Buzzer;

Buzzer\_Delay500us();

}

}

十五、红外遥控

**IR.h**

#ifndef \_\_IR\_H\_\_

#define \_\_IR\_H\_\_

#define IR\_POWER 0x45

#define IR\_MODE 0x46

#define IR\_MUTE 0x47

#define IR\_START\_STOP 0x44

#define IR\_PREVIOUS 0x40

#define IR\_NEXT 0x43

#define IR\_EQ 0x07

#define IR\_VOL\_MINUS 0x15

#define IR\_VOL\_ADD 0x09

#define IR\_0 0x16

#define IR\_RPT 0x19

#define IR\_USD 0x0D

#define IR\_1 0x0C

#define IR\_2 0x18

#define IR\_3 0x5E

#define IR\_4 0x08

#define IR\_5 0x1C

#define IR\_6 0x5A

#define IR\_7 0x42

#define IR\_8 0x52

#define IR\_9 0x4A

void IR\_Init(void);

unsigned char IR\_GetDataFlag(void);

unsigned char IR\_GetRepeatFlag(void);

unsigned char IR\_GetAddress(void);

unsigned char IR\_GetCommand(void);

#endif

**IR.c**

#include <REGX52.H>

#include "Timer0.h"

#include "Int0.h"

unsigned int IR\_Time;

unsigned char IR\_State;

unsigned char IR\_Data[4];

unsigned char IR\_pData;

unsigned char IR\_DataFlag;

unsigned char IR\_RepeatFlag;

unsigned char IR\_Address;

unsigned char IR\_Command;

/\*\*

\* @brief 红外遥控初始化

\* @param 无

\* @retval 无

\*/

void IR\_Init(void)

{

Timer0\_Init();

Int0\_Init();

}

/\*\*

\* @brief 红外遥控获取收到数据帧标志位

\* @param 无

\* @retval 是否收到数据帧，1为收到，0为未收到

\*/

unsigned char IR\_GetDataFlag(void)

{

if(IR\_DataFlag)

{

IR\_DataFlag=0;

return 1;

}

return 0;

}

/\*\*

\* @brief 红外遥控获取收到连发帧标志位

\* @param 无

\* @retval 是否收到连发帧，1为收到，0为未收到

\*/

unsigned char IR\_GetRepeatFlag(void)

{

if(IR\_RepeatFlag)

{

IR\_RepeatFlag=0;

return 1;

}

return 0;

}

/\*\*

\* @brief 红外遥控获取收到的地址数据

\* @param 无

\* @retval 收到的地址数据

\*/

unsigned char IR\_GetAddress(void)

{

return IR\_Address;

}

/\*\*

\* @brief 红外遥控获取收到的命令数据

\* @param 无

\* @retval 收到的命令数据

\*/

unsigned char IR\_GetCommand(void)

{

return IR\_Command;

}

//外部中断0中断函数，下降沿触发执行

void Int0\_Routine(void) interrupt 0

{

if(IR\_State==0) //状态0，空闲状态

{

Timer0\_SetCounter(0); //定时计数器清0

Timer0\_Run(1); //定时器启动

IR\_State=1; //置状态为1

}

else if(IR\_State==1) //状态1，等待Start信号或Repeat信号

{

IR\_Time=Timer0\_GetCounter(); //获取上一次中断到此次中断的时间

Timer0\_SetCounter(0); //定时计数器清0

//如果计时为13.5ms，则接收到了Start信号（判定值在12MHz晶振下为13500，在11.0592MHz晶振下为12442）

if(IR\_Time>12442-500 && IR\_Time<12442+500)

{

IR\_State=2; //置状态为2

}

//如果计时为11.25ms，则接收到了Repeat信号（判定值在12MHz晶振下为11250，在11.0592MHz晶振下为10368）

else if(IR\_Time>10368-500 && IR\_Time<10368+500)

{

IR\_RepeatFlag=1; //置收到连发帧标志位为1

Timer0\_Run(0); //定时器停止

IR\_State=0; //置状态为0

}

else //接收出错

{

IR\_State=1; //置状态为1

}

}

else if(IR\_State==2) //状态2，接收数据

{

IR\_Time=Timer0\_GetCounter(); //获取上一次中断到此次中断的时间

Timer0\_SetCounter(0); //定时计数器清0

//如果计时为1120us，则接收到了数据0（判定值在12MHz晶振下为1120，在11.0592MHz晶振下为1032）

if(IR\_Time>1032-500 && IR\_Time<1032+500)

{

IR\_Data[IR\_pData/8]&=~(0x01<<(IR\_pData%8)); //数据对应位清0

IR\_pData++; //数据位置指针自增

}

//如果计时为2250us，则接收到了数据1（判定值在12MHz晶振下为2250，在11.0592MHz晶振下为2074）

else if(IR\_Time>2074-500 && IR\_Time<2074+500)

{

IR\_Data[IR\_pData/8]|=(0x01<<(IR\_pData%8)); //数据对应位置1

IR\_pData++; //数据位置指针自增

}

else //接收出错

{

IR\_pData=0; //数据位置指针清0

IR\_State=1; //置状态为1

}

if(IR\_pData>=32) //如果接收到了32位数据

{

IR\_pData=0; //数据位置指针清0

if((IR\_Data[0]==~IR\_Data[1]) && (IR\_Data[2]==~IR\_Data[3])) //数据验证

{

IR\_Address=IR\_Data[0]; //转存数据

IR\_Command=IR\_Data[2];

IR\_DataFlag=1; //置收到连发帧标志位为1

}

Timer0\_Run(0); //定时器停止

IR\_State=0; //置状态为0

}

}

}

十六、矩阵键盘

**MatrixKey.h**

#ifndef \_\_MATRIXKEY\_H\_\_

#define \_\_MATRIXKEY\_H\_\_

unsigned char MatrixKey();

#endif

**MatrixKey.c**

#include <REGX52.H>

#include "Delay.h"

/\*\*

\* @brief 矩阵键盘读取按键键码

\* @param 无

\* @retval KeyNumber 按下按键的键码值

如果按键按下不放，程序会停留在此函数，松手的一瞬间，返回按键键码，没有按键按下时，返回0

\*/

unsigned char MatrixKey()

{

unsigned char KeyNumber = 0;

P1 = 0xFF; P1\_3 = 0;

if (P1\_7 == 0) { Delay(20); while (P1\_7 == 0); Delay(20); KeyNumber = 1; }

if (P1\_6 == 0) { Delay(20); while (P1\_6 == 0); Delay(20); KeyNumber = 5; }

if (P1\_5 == 0) { Delay(20); while (P1\_5 == 0); Delay(20); KeyNumber = 9; }

if (P1\_4 == 0) { Delay(20); while (P1\_4 == 0); Delay(20); KeyNumber = 13; }

P1 = 0xFF; P1\_2 = 0;

if (P1\_7 == 0) { Delay(20); while (P1\_7 == 0); Delay(20); KeyNumber = 2; }

if (P1\_6 == 0) { Delay(20); while (P1\_6 == 0); Delay(20); KeyNumber = 6; }

if (P1\_5 == 0) { Delay(20); while (P1\_5 == 0); Delay(20); KeyNumber = 10; }

if (P1\_4 == 0) { Delay(20); while (P1\_4 == 0); Delay(20); KeyNumber = 14; }

P1 = 0xFF; P1\_1 = 0;

if (P1\_7 == 0) { Delay(20); while (P1\_7 == 0); Delay(20); KeyNumber = 3; }

if (P1\_6 == 0) { Delay(20); while (P1\_6 == 0); Delay(20); KeyNumber = 7; }

if (P1\_5 == 0) { Delay(20); while (P1\_5 == 0); Delay(20); KeyNumber = 11; }

if (P1\_4 == 0) { Delay(20); while (P1\_4 == 0); Delay(20); KeyNumber = 15; }

P1 = 0xFF; P1\_0 = 0;

if (P1\_7 == 0) { Delay(20); while (P1\_7 == 0); Delay(20); KeyNumber = 4; }

if (P1\_6 == 0) { Delay(20); while (P1\_6 == 0); Delay(20); KeyNumber = 8; }

if (P1\_5 == 0) { Delay(20); while (P1\_5 == 0); Delay(20); KeyNumber = 12; }

if (P1\_4 == 0) { Delay(20); while (P1\_4 == 0); Delay(20); KeyNumber = 16; }

return KeyNumber;

}

十七、人体红外传感器

说明人体红外传感器在有物体遮挡时输出高电平，否则输出低电平；

将传感器输出引脚接单片机I/O端口，检测端口电平变化，当端口电平为高电平时说明传感器检测到物体，液晶屏显示“Y”，

否则单片机端口输出低电平，液晶屏显示“N”。

**Body\_hw.c**

sbit hw = P1 ^ 0; //定义红外接口

void main()

{

uchar i = 0;

Delay(10); //延时

LCD12864\_Init(); //初始化LCD

LCD12864\_Position(0, 0); //设置显示位置为第一行的第1个字符

while (dis1[i] != '\0') //显示字符

LCD12864\_WriteData(dis1[i++]);

while (1)

{

LCD12864\_Position(0, 2);

LCD12864\_WriteData(':');

if (hw == 0)

LCD12864\_WriteData('N');

else

LCD12864\_WriteData('Y');

Delay(5);

}

}

十八、延时函数

**delay.h**

#ifndef \_delay\_H

#define \_delay\_H

#include<reg52.h>

void delay\_ms (u16 ms);

void delay\_us (u16 ten\_us);

#endif

**delay.c**

#include "delay.h"

void delay\_ms(u16 ms) //：延时函数,延时1ms

{

unsigned char i, j;

i = 2;

j = 199;

do

{

while (--j);

} while (--i);

}

void delay\_us(u16 ten\_us) //:延时10us的函数

{

unsigned char i;

i = 2;

while (--i);

}

十九、中断、定时器

**Timer0.h**

#ifndef \_\_TIMER0\_H\_\_

#define \_\_TIMER0\_H\_\_

#include "reg52.h"

unsigned int T0Count;

/\*\*

\* @brief 定时器0初始化，1毫秒@12.000MHz

\* @param 无

\* @retval 无

\*/

void Timer0\_Init(void)

{

TMOD &= 0xF0; //设置定时器模式

TMOD |= 0x02; //设置定时器模式

TL0 = 0xA4; //设置定时初始值

TH0 = 0xA4; //设置定时重载值

TF0 = 0; //清除TF0标志

TR0 = 0; //定时器0开始计时

ET0 = 1;

EA = 1;

PT0 = 0;

}

//定时器中断函数

void Timer0\_Routine() interrupt 1

{

TL0 = 0xA4; //设置定时初值

TH0 = 0xA4; //设置定时初值

T0Count++;

}

#endif

**Timer1.h**

#ifndef \_\_TIMER1\_H\_\_

#define \_\_TIMER1\_H\_\_

#include <REGX52.H>

unsigned int T1Count;

/\*\*

\* @brief 定时器1初始化，100us@12.000MHz

\* @param 无

\* @retval 无

\*/

void Timer1\_Init(void)

{

TMOD &= 0x0F; //设置定时器模式

TMOD |= 0x20; //设置定时器模式

TL1 = 0xA4; //设置定时初始值

TH1 = 0xA4; //设置定时重载值

TF1 = 0; //清除TF1标志

TR1 = 1; //定时器1开始计时

ET1 = 1;

EA = 1;

PT1 = 0;

}

//定时器中断函数模板

void Timer1\_Routine() interrupt 3

{

static unsigned int T1Count;

TL1 = 0x9C; //设置定时初值

TH1 = 0xFF; //设置定时初值

T1Count++;

if(T1Count>=1000)

T1Count=0;

}

#endif

**uart.h**

#ifndef \_\_UART\_H\_\_

#define \_\_UART\_H\_\_

#include<reg52.h>

void UART1\_Init(void);

void UART1\_SendByte(unsigned char DAT);

void UART1\_SendString(unsigned char \*DAT, unsigned char len);

/\*串口1初始化\*/

void UART1\_Init(void)

{

SCON = 0x50; //8位数据,可变波特率

TMOD|= 0x20; //设定定时器1为8位自动重装方式

TL1 = 0xFD; //设定定时初值

TH1 = 0xFD; //设定定时器重装值

ET1 = 0; //禁止定时器1中断

TR1 = 1; //启动定时器1

ES = 1;

EA = 1;

RI = 0;

TI = 0;

}

/\*串口发送一字节数据\*/

void UART1\_SendByte(unsigned char DAT)

{

SBUF = DAT;

while(TI == 0);

TI=0;

}

/\*串口发送字符串数据\*/

void UART1\_SendString(unsigned char \*DAT, unsigned char len)

{

unsigned char i;

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

{

UART1\_SendByte(\*DAT++);

}

}

#endif

二十、其他

**main.h**

#ifndef \_main\_H

#define \_main\_H

typedef unsigned char u8; /\* defined for unsigned 8-bits integer variable 无符号8位整型变量 \*/

typedef signed char S8; /\* defined for signed 8-bits integer variable 有符号8位整型变量 \*/

typedef unsigned int u16; /\* defined for unsigned 16-bits integer variable 无符号16位整型变量 \*/

typedef signed int S16; /\* defined for signed 16-bits integer variable 有符号16位整型变量 \*/

typedef unsigned long U32; /\* defined for unsigned 32-bits integer variable 无符号32位整型变量 \*/

typedef signed long S32; /\* defined for signed 32-bits integer variable 有符号32位整型变量 \*/

typedef float F32; /\* single precision floating point variable (32bits) 单精度浮点数（32位长度） \*/

typedef double F64; /\* double precision floating point variable (64bits) 双精度浮点数（64位长度） \*/

#endif