

1 目录

| | | |
|------|--------------|----|
| 2 | SYSTEM\SYS | 2 |
| 2.1 | SYS.C | 2 |
| 2.2 | SYS.H | 3 |
| 3 | AD | 4 |
| 3.1 | AD.C | 4 |
| 3.2 | AD.H | 5 |
| 4 | CSBCJ | 5 |
| 4.1 | CSBCJ.C | 5 |
| 4.2 | CSBCJ.H | 6 |
| 5 | DHT11 | 6 |
| 5.1 | DHT11.C | 6 |
| 5.2 | DHT11.H | 9 |
| 6 | DS18B20 | 9 |
| 6.1 | DS18B20.C | 9 |
| 6.2 | DS18B20.H | 12 |
| 7 | DS1302 | 12 |
| 7.1 | DS1302.C | 12 |
| 7.2 | DS1302.H | 15 |
| 8 | eeeprom | 16 |
| 8.1 | 24cxx.c | 16 |
| 8.2 | 24cxx.h | 18 |
| 8.3 | i2c.c | 18 |
| 8.4 | i2c.h | 21 |
| 9 | HX711 | 21 |
| 9.1 | HX711.C | 21 |
| 9.2 | HX711.H | 22 |
| 10 | KEY | 23 |
| 10.1 | KEY.c | 23 |
| 10.2 | KEY.H | 25 |
| 11 | LCD12864 | 25 |
| 11.1 | LCD12864.C | 25 |
| 11.2 | LCD12864.H | 30 |
| 11.3 | LCD12864 字符表 | 32 |
| 12 | MOTOR | 32 |
| 12.1 | motor.c | 32 |
| 12.2 | motor.h | 34 |
| 13 | MPU6050 | 34 |
| 13.1 | MPU6050.C | 34 |
| 13.2 | MPU6050.H | 38 |
| 14 | TIMER | 39 |
| 14.1 | TIMER.C | 39 |
| 14.2 | TIMER.H | 41 |

2 SYSTEM\SYS

2.1 SYS.C

```
#include "sys.h"

/*
24MHz 主频下
在示波器下看
一个语句的时间大概是 0.8us
*/
//2 微秒 延时
//此函数是以 2us 的倍数增加
//t 2us 的倍数
void delay2us(u16 us)
{
    u16 i;
    u8 m;
    for(i=0;i<us;i++)
        for(m=0;m<3;m++);
}

void delay_ms(u16 ms)
{
    u16 m;
    for(m=0;m<ms;m++)
        delay4us(250);
}

//为兼容原程序
void delay_us(u16 us)
{
    u16 i;
    u8 m;
    for(i=0;i<us;i++)
        for(m=0;m<1;m++);
}

//4 微秒延时
//此函数是以 4us 的倍数增加
//t 4us 的倍数
void delay4us(u16 us)
{
    u16 i;
    u8 m;
    for(i=0;i<us;i++)
        for(m=0;m<5;m++);
}
```

```

}

//1 毫秒 延时
//ms 1ms 的倍数
void delay1ms(u16 ms)
{
    u16 m;
    for(m=0;m<ms;m++)
        delay4us(250);
}

void Delay100ms(void) //@24.000MHz
{
    unsigned char i, j, k;

    _nop_();
    _nop_();
    i = 10;
    j = 31;
    k = 147;
    do
    {
        do
        {
            while (--k);
        } while (--j);
    } while (--i);
}

```

2.2 SYS.H

```

#ifndef __SYS_H
#define __SYS_H

#include "stc15f2k60s2.h"
#include <intrins.h>

#define DRIVING_SIMULATION 0 //是否是定时器脉冲模拟计价/码盘计数

#define u8 unsigned char
#define u16 unsigned int
#define u32 unsigned long

void delay2us(u16 us); //2us 倍数延时
void delay4us(u16 us); //4us 倍数延时
void delay1ms(u16 ms); //毫秒延时

```

```

void delay_us(u16 us);    //为兼容源程序
void delay_ms(u16 ms);
void Delay100ms(void);
#endif

```

3 AD

3.1 AD.C

```

#include "stc15f2k60s2.h"
#include "sys.h"
#include "intrins.h"
#include "ad.h"

/*Define ADC operation const for ADC_CONTR*/
#define ADC_POWER    0x80        //ADC power control bit
#define ADC_FLAG     0x10        //ADC complete flag    模数转换结束标志位
#define ADC_START     0x08        //ADC start control bit  模数转换启动控制位

//转换速度控制位 SPEED0 和 SPEED1，共四种状态，对应四种转换速度
#define ADC_SPEEDLL  0x00        //540 clocks
#define ADC_SPEEDL   0x20        //360 clocks
#define ADC_SPEEDH   0x40        //180 clocks
#define ADC_SPEEDHH  0x60        //90 clocks

/*-----
Get ADC result
-----*/
u16 GetADCResult(u8 ch)          //
{
    u16 Vo;

    P1ASF = 0x01;                //选择 P1 口的哪一口 这里的口和 ch 要对应才能达到选择该口
    ADC_CONTR = ADC_POWER | ADC_SPEEDLL | ch | ADC_START; //0x00|0x00|ch|0x08:选择 A/D 输入通道，开始 A/D 转换
    // 这么用语句的主要原因就是不能位寻址
    // 通道选择在后 3 位所以直接用一个整数表示 ch
    //例如 ch=6 那么对应的后三位就是 110

    _nop_();                      //Must wait before inquiry ,
    _nop_();                      //设置 ADC_CONTR 寄存器后需加 4 个 CPU 时钟周期的延时，才能保证值被写入 ADC_CONTR 寄存器
    _nop_();
    _nop_();
    while (!(ADC_CONTR & ADC_FLAG)); //Wait complete flag

    ADC_CONTR &= ~ADC_FLAG;        //Close ADC 将标志位清零等待下次硬件置 1
    //也可以写成 ADC_CONTR= ADC_CONTR & (~ADC_FLAG)

    Vo=(ADC_RES<<2)+ADC_RESL;      //打开 10 位 AD 采集功能 如果用 8 位 AD Vo=ADC_RESL 即可
    //10 位 AD 采集 即 2 的 10 次方 满值为 1024 这里用 1024 表示 5 伏的电压
    //8 位 AD 采集 即 2 的 8 次方 满值为 256 用 256 表示 5 伏

    return Vo;
}

```

```
/******  
/
```

3.2 AD.H

```
#ifndef AD_H  
#define AD_H  
#include "sys.h"  
  
u16 GetADCResult(u8 ch); //读取 P1.0 口 AD 值  
  
#endif
```

4 CSBCJ

4.1 CSBCJ.C

```
#include<reg52.h>  
sbit Trig = P2^0;  
sbit Echo = P2^1;  
void Delay10us()  
{  
    TMOD |= 0x1;  
    TH0 = 0xFF;  
    TL0 = 0xF6;  
    TR0 = 1;  
    while(!TF0);  
    TF0 = 0;  
}  
void Rstart()  
{  
    Trig=0;  
    Trig=1;  
    Delay10us();  
    Trig=0;  
}  
int gettime()  
{  
    unsigned int time = 0;  
    time = TH0*256+TL0;// TH0<<8 | TL0;  
    return time;  
}  
float Getdis(unsigned int time)  
{  
    float distance;  
    distance = (float)time * 0.017;  
    TH0=0;  
    TL0=0;  
    return distance;  
}  
void star()  
{  
    TH0 = 0;
```

```

    TL0 = 0;
    TR0 = 1;
}
void end()
{
    TR0 = 0;
}
void GetOnce()
{
    Rstart();
    while(!Echo);
    star();
    while(Echo);
    end();
}
void delay()
{
    int i;
    int j;
    for(i=0;i<100;i++)
        for(j = 0;j<2000;j++);
}

```

4.2 CSBCJ.H

```

#ifndef __XPT2046_H__
#define __XPT2046_H__
void Delay10us();
void Rstart() ;
int gettime();
float Getdis(unsigned int time) ;
void star();
void end();
void GetOnce();
void delay();
#endif

```

5 DHT11

5.1 DHT11.C

```

#include "stc15f2k60s2.h"
#include <stdio.h>
#include "string.h"
#include "sys.h"

sbit DHT_IO = P1^1 ;

u8 U8FLAG,k;
u8 U8count,U8temp;

```

```

u8  U8T_data_H,U8T_data_L,U8RH_data_H,U8RH_data_L,U8checkdata;
u8  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;

```

```

void Delay_10us(void)

```

```

{
    delay_us(10);
}

```

```

void COM(void)

```

```

{

    u8 i;

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

        U8FLAG=2;
        while((!DHT_IO)&&U8FLAG++);
        Delay_10us();
        Delay_10us();
        Delay_10us();
        U8temp=0;
        if(DHT_IO)U8temp=1;
        U8FLAG=2;
        while((DHT_IO)&&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;        //0
    }//rof
}

```

```

//-----

```

```

//-----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-----
//---- delay1ms();, Delay_10us();,COM();
//-----

void UpdateTemp(void)
{
    //主机拉低 18ms
    DHT_IO=0;
    delay1ms(18);
    DHT_IO=1;
    //总线由上拉电阻拉高 主机延时 20us
    Delay_10us();
    Delay_10us();
//    Delay_10us();
//    Delay_10us();
    //主机设为输入 判断从机响应信号
    DHT_IO=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
    if(!DHT_IO)        //T !
    {
        U8FLAG=2;
        //判断从机是否发出 80us 的低电平响应信号是否结束
        while((!DHT_IO)&&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((DHT_IO)&&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;
        DHT_IO=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;
    }//fi
}

}

void show_dht11(){
    UpdateTemp();
    print_n(0,0,U8RH_data_H,2);
    print(".");
    print_n(0,3,U8RH_data_L,2);
}

```

5.2 DHT11.H

```

#ifndef DHT11_H
#define DHT11_H

#include "sys.h"
#include "stc15f2k60s2.h"

extern u8 U8T_data_H,U8T_data_L,U8RH_data_H,U8RH_data_L,U8checkdata;

void COM(void);
void UpdateTemp(void);
void show_dht11();
#endif

```

6 DS18B20

6.1 DS18B20.C

```

#include "stc15f2k60s2.h"
#include "sys.h"
#include "ds18b20.h"
#include "gui.h"

sbit DQ=P1^3;    //ds18b20 端口

//DS18B20 复位函数
void ow_reset(void)
{
    DQ=1;        //从高拉倒低
    DQ=0;
    delay4us(125);    //>480 us    低电平复位信号
}

```

```

    DQ=1;
    delay4us(4);          //>15us 的上升沿      15-60us 高电平后 是 60-240us 的应答信号
}

//等待 DS18B20 的回应
//返回 1:未检测到 DS18B20 的存在
//返回 0:存在
u8 ds18B20_check(void)
{
    u8 retry=0;          //检测计算变量
    while (DQ&&retry<50)
    {
        retry++;
        delay4us(1);     //大概 4us
    };
    if(retry>=100)return 1;
    else retry=0;
    while (!DQ&&retry<60)    //保持 240us 的延时 完成整个的复位过程
    {
        retry++;
        delay4us(1);     //大概 4us
    };

    if(retry>=60)return 1;    //没有接到 DS18B20 应答
    return 0;                //接到应答
}

//初始化 DS18B20 同时检测 DS 的存在
//返回 1:不存在
//返回 0:存在
u8 ds18b20_init(void)
{
    u8 m;
    ow_reset();            //复位总线
    m=ds18B20_check();    //等等 ds 应答
    return m;
}

//DS18B20 写命令函数
//向 1-WIRE 总线上写 1 个字节
//u8 val 要写入字节
//DS18B20 手册最下面有时序图
void write_byte(u8 val)
{
    u8 i,testb;

```

```

for(i=8;i>0;i--)
{
    testb=val&0x01;          //最低位移出
    val=val>>1;
    if(testb)                  //写 1
    {
        DQ=0;
        delay4us(1);          //4us
        DQ=1;
        delay4us(15);          //60us
    }
    else                        //写 0
    {
        DQ=0;
        delay4us(15);          //60us
        DQ=1;
        delay4us(1);           //4us
    }
}
}

//DS18B20 读 1 字节函数
//从总线上取 1 个字节
//返回值为读取字节值
//说明 一次 1bit 的读取最少需要 60us 两次读取之间需要至少 1us 的恢复时间
// 单次读取 1bit 总线拉低不能超过 15us 然后马上拉高
u8 read_byte(void)
{
    u8 i;
    u8 value=0;
    for(i=8;i>0;i--)
    {
        DQ=1;
        value>>=1; //value=value>>1
        DQ=0;
        delay4us(2); //拉低 4us
        DQ=1; //拉高
        delay4us(3); //拉高 10us 准备接收总线当前数据
        if(DQ)value|=0x80; //将当前数据值存入临时变量
        delay4us(13); //50 us 延时 完成一次读取的延时（一次读取最少 60us）跳过 1us 的恢复时间
    }
    DQ=1;
    return(value);
}

//读出温度函数
//返回为温度值 温度值为 short 变量 有正负

```

```

short read_temp()  //short 可以表示-32768~+32767
{
    u8 TL,TH;
    u8 temp;
    short t;
    ow_reset();          //总线复位
    ds18B20_check();      //等待 DS 应答
    write_byte(0xcc);     //发命令
    write_byte(0x44);     //发转换命令
    ow_reset();          //复位
    ds18B20_check();      //等待 DS 应答
    write_byte(0xcc);     //发命令
    write_byte(0xbe);     //发送读温度命令
    TL=read_byte();       //读温度值的低字节
    TH=read_byte();       //读温度值的高字节
    t=TH;

    if(TH>7)
    {
        TH=~TH;
        TL=~TL;
        temp=0;//温度为负
    }else temp=1;
    t<=8;
    t+=TL;                // 两字节合成一个整型变量。
    t=(float)t*0.625;     //0.0625 为 12 位温度采集的分辨率      t 为采集的数值 这里扩大 10 倍提取小数点后一位
    if(temp)return t;
    else return -t;
}

```

6.2 DS18B20.H

```

#ifndef DS18B20_H
#define DS18B20_H

short read_temp();
#endif

```

7 DS1302

7.1 DS1302.C

```

//DS1302 部分
//整体 显示 时间 日期 年月 星期
//可不停更新秒时间
#include "stc15f2k60s2.h"
#include "ds1302.h"
#include "intrins.h"
#include "sys.h"
#include "lcd12864.h"

code u8 write_rtc_address[7]={0x80,0x82,0x84,0x86,0x88,0x8a,0x8c}; //秒分时日月周年 最低位读写位

```

```
code u8 read_rtc_address[7]={0x81,0x83,0x85,0x87,0x89,0x8b,0x8d};
```

```
u8 ds1302tmp[7]={50,03,02,20,5,6,23}; //秒分日月周年
```

2020 年 4 月 20 日 8 点 20 分 星期日 这个数是自己设定的

```
//写一个字节
```

```
//temp 要写入的字节 （地址或数据）
```

```
void Write_Ds1302_Byte(u8 temp)
```

```
{
    u8 i;
    for (i=0;i<8;i++)    //循环 8 次 写入数据
    {
        SCK=0;
        SDA=temp&0x01;    //每次传输低字节
        temp>>=1;        //右移一位
        SCK=1;
    }
}
```

```
//写入 DS1302 数据
```

```
//address 写入的地址
```

```
//dat 写入的数据
```

```
void Write_Ds1302( u8 address,u8 dat )
```

```
{
    CE=0;
    _nop_();
    SCK=0;
    _nop_();
    CE=1;                //启动
    _nop_();
    Write_Ds1302_Byte(address); //发送地址
    Write_Ds1302_Byte(dat);     //发送数据
    CE=0;                    //恢复
}
```

```
//读出 DS1302 数据
```

```
//address 读取数据的地址
```

```
//返回 读取的数据值
```

```
u8 Read_Ds1302 ( u8 address )
```

```
{
    u8 i,temp=0x00;
    CE=0;
    _nop_();
    _nop_();
    SCK=0;
    _nop_();
    _nop_();
    CE=1;
    _nop_();
    _nop_();

    Write_Ds1302_Byte(address);
```

```

for (i=0;i<8;i++)      //循环 8 次 读取数据
{
    if(SDA)
        temp|=0x80;      //每次传输低字节
    SCK=0;
    temp>>=1;      //右移一位
    _nop_();
    _nop_();
    _nop_();
    SCK=1;
}

CE=0;
_nop_();      //以下为 DS1302 复位的稳定时间
_nop_();
CE=0;
SCK=0;
_nop_();
_nop_();
_nop_();
_nop_();
SCK=1;
_nop_();
_nop_();
SDA=0;
_nop_();
_nop_();
SDA=1;
_nop_();
_nop_();
return (temp);      //返回
}

//写入时 DCB 转换
//DS1302 只接收 DCB 码 即高 4 位放 10 位值 低 4 位放个位值
//add 要转换的地址
//返回 转换后的数据
u8 bcd_read(u8 add)
{
    u8 fla,fla2;
    fla=Read_Ds1302(add);
    fla2=((fla/16)*10)+(fla&0x0f);
    return fla2;
}

//设定时钟数据
//主要设置时钟芯片里的 秒分时日月周年
void Set_RTC(void)      //设定 日历

```

```

{
    u8 i;
    for(i=0;i<7;i++)          //BCD 处理
        ds1302tmp[i]=ds1302tmp[i]/10*16+ds1302tmp[i]%10;  //ds1302tmp[7]={0,20,8,20,4,7,20};

    Write_Ds1302(0x8E,0x00);  //写使能

    for(i=0;i<7;i++)          //7 次写入 秒分时日月周年
        Write_Ds1302(write_rtc_address[i],ds1302tmp[i]);

    Write_Ds1302(0x8E,0x80);  //写禁止
}

//DS1302 时间 显示函数
//此函数需要循环 进行更新 不能自动更新
//例 20:25
void ds1302_scan(void)
{
    u8 temp;

    temp = bcd_read(0x85);//时

    LCD12864_SetWindow(3,0);  //第 4 行    显示
    LCD12864_WriteData(temp/10%10+0x30);
    LCD12864_WriteData(temp%10+0x30);
    LCD12864_WriteData(':');

    temp = bcd_read(0x83);//分

    LCD12864_WriteData(temp/10%10+0x30);
    LCD12864_WriteData(temp%10+0x30);

    temp = bcd_read(0x81);//秒
    LCD12864_WriteData(':');
    LCD12864_WriteData(temp/10%10+0x30);
    LCD12864_WriteData(temp%10+0x30);

    print("你好! ");
}

```

7.2 DS1302.H

```

#ifndef __DS1302_H
#define __DS1302_H
#include "sys.h"

sbit SCK=P3^5;    //B
sbit SDA=P3^6;    //
sbit CE=P5^4;     //

```

```

//void Set_RTC(void);    //set RTC
void ds1302_scan(void); //读数据

void Write_Ds1302(u8 address,u8 dat);
u8 bcd_read(u8 add);
u8 Read_Ds1302 ( u8 address );
void Set_RTC();

#endif

```

8 EEPROM

8.1 24cxx.c

```

/*****此部分为 AT2402 的驱动程序使用 I2C 总线连接*****/
#include "stc15f2k60s2.h"
#include "sys.h"
#include "gui.h"
#include "i2c.h"
#include "24cxx.h"

//24c 默认的硬件地址都是 0xa0 即 1010 然后 A2 A1 A0 最后一位 为读写位 1 读 0 写
//24c02 256x8 bit
//24c04 512x8 bit
//而子地址 只是 8 位的 即最大也就 256 个字节 那么 24c04 就要占用 2 个 256 个字节
//这里 24c04 硬件上采用的是 分页方法 即有 24c02 里有 A0 A1 A2 当是 24c04 时 A0 做为分页位
//当 A0 为 0 表示第一页 写入第一页 256 字节
//当 A0 为 1 表示第二页 写入第二页 256 字节

//修改为 24c256!!!!!!!!!!!!

//在 AT24CXX 指定地址读出一个数据
//Addr: 开始读数的地址
//返回值 : 读到的数据
u8 AT24C_Rcvone(u16 Addr)
{
    u8 temp=0;
    Start_I2c();           //启动总线
    SendByte(0xa0);        //发送写命令
    I2c_wait_ack();        //等待应答
    SendByte(Addr>>8);      //发送高地址
    I2c_wait_ack();        //等待应答

    SendByte(Addr%256);     //发送低地址
    I2c_wait_ack();        //等待应答
}

```



```

Start_I2c();           //重新启动总线
SendByte(0xa1);        //设置为读操作
I2c_wait_ack();        //等待应答;

temp=RcvByte();        //读字节
Ack_I2c(0);            //非应答   //???

Stop_I2c();            //结束总线
return temp;
}

```

//在 AT24CXX 指定地址写入一个数据 此函数只限于 c256 其他未做验证

//Addr: 写入数据的目的地址

//Data: 要写入的数据

```
void AT24C_Sendone(u16 Addr,u8 Data)
```

```

{
    Start_I2c();        //启动总线

    SendByte(0xa0);      //发送写命令
    I2c_wait_ack();      //等待应答
    SendByte(Addr>>8);   //发送高地址
    I2c_wait_ack();      //等待应答
    SendByte(Addr%256);  //发送低地址
    I2c_wait_ack();      //等待应答
    SendByte(Data);      //发送字节数据
    I2c_wait_ack();      //等待应答
    Stop_I2c();          //结束总线
    delay1ms(10);        //如果是连续发送字节的时候这个延时很重要 否则将回传错
}

```

//在 AT24CXX 里面的指定地址开始写入长度为 Len 的数据

//该函数用于写入 16bit 或者 32bit 的数据。

//Addr : 开始写入的地址

//Data : 数据数组首地址

//Len : 要写入数据的长度 2,4

```
void AT24C_SendLenByte(u16 Addr,u8 *Data,u8 Len)
```

```

{
    while(Len--)
    {
        AT24C_Sendone(Addr,*Data);
        Addr++;
        Data++;
    }
}

```

```

//在 AT24CXX 里面的指定地址开始读出长度为 Len 的数据
//该函数用于读出 16bit 或者 32bit 的数据。
//Addr    :开始读出的地址
//返回值   :数据
//Len      :要读出数据的长度 2,4
void AT24C_RcvLenByte(u16 Addr,u8 *temp,u8 Len)
{
    while(Len)
    {
        *temp++=AT24C_Rcvone(Addr++);
        Len--;
    }
}

```

8.2 24CXX.H

```

#ifndef Z24C02_H
#define Z24C02_H
#include "sys.h"

//void c02();                                //24c 测试函数
void AT24C_Sendone(u16 Addr,u8 Data);        //写一个字节
u8 AT24C_Rcvone(u16 Addr);                   //读一个字节
void AT24C_SendLenByte(u16 Addr,u8 *Data,u8 Len); //写多个字节
void AT24C_RcvLenByte(u16 Addr,u8 *temp,u8 Len); //读多个字节
#endif

```

8.3 I2C.c

```

/*****此部分为 I2C 总线的驱动程序*****/

#include "stc15f2k60s2.h"
#include "i2c.h"
#include "sys.h"

sbit SCL=P6^7; //I2C 时钟
sbit SDA=P6^6; //I2C 数据

//SDA 方向控制
void IO_SDA(u8 m)
{
    if(m==1)
    {
        P6M1 |= (1<<6); //配置 SDA 为输入 1 0
        P6M0 &= ~(1<<6);
    }
}

```

```

}
else
{
P6M1 &=~(1<<6);           //恢复 SDA 双向 io    0 0
P6M0 &=~(1<<6);
}
}
}

```

//起动总线函数

//Start_I2c();

//功能: 启动 I2C 总线,即发送 I2C 起始条件.

void Start_I2c()

```

{
SDA=1;           //发送起始条件的数据信号
SCL=1;
delay4us(2);
SDA=0;           //发送起始信号
delay4us(2);
SCL=0;           //钳住 I2C 总线,准备发送或接收数据
}

```

//结束总线函数

//Stop_I2c();

//功能: 结束 I2C 总线,即发送 I2C 结束条件.

void Stop_I2c()

```

{
SCL=0;
SDA=0;           //发送结束条件的数据信号
delay4us(2);
SCL=1;           //结束条件建立时间大于 4μs
SDA=1;           //发送 I2C 总线结束信号
delay4us(2);
}

```

//等待应答信号到来

//返回值: 1, 接收应答失败

// 0, 接收应答成功

u8 I2c_wait_ack(void)

```

{
u8 Time=0;
IO_SDA(1);           //配置 SDA 为输入
SDA=1;               //准备接收应答位
delay4us(1);
SCL=1;
delay4us(1);
while(SDA)

```

```

{
    Time++;
    if(Time>250)
    {
        Stop_I2c();
        IO_SDA(0);    //恢复 SDA 双向 io
        return 1; //无应答返回 1
    }
}

SCL=0;//时钟输出 0

IO_SDA(0);          //恢复 SDA 双向 io
return 0;            //有应答返回 0
}

//字节数据发送函数
//SendByte(u8 c);
//功能：    将数据 c 发送出去,可以是地址,也可以是数据
void SendByte(u8 c)
{
    u8 BitCnt;

                                //条件 一定要开启总线 保持 SCL 处于 0 状态 才能进行写入
    for(BitCnt=0;BitCnt<8;BitCnt++) //要传送的数据长度为 8 位
    {
        if((c<<BitCnt)&0x80)SDA=1;    //判断发送位 发送是由高位开始发送
        else SDA=0;
        delay4us(1);
        SCL=1;                        //置时钟线为高,通知被控器开始接收数据位
        delay4us(1);
        SCL=0;
        delay4us(1);
    }
}

//字节数据接收函数
//RcvByte();
//功能：    用来接收从器件传来的数据,并判断总线错误(不发应答信号),
//          发完后请用应答函数应答从机。
u8 RcvByte()
{
    u8 retc=0,
        i;
    IO_SDA(1);                        //配置 SDA 为输入

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

```

```

{
    SCL=0;                //置时钟线为低, 准备接收数据位
    delay4us(1);
    SCL=1;                //置时钟线为高使数据线上数据有效
    retc<=<=1;
    if(SDA==1)retc++;      //读数据位,接收的数据位放入 retc 中
    delay4us(1);
}
IO_SDA(0);                //恢复 SDA 双向 io
return retc;
}

//应答子函数
//Ack_I2c(bit a);
//功能:主控器进行应答信号(可以是应答或非应答信号, 由位参数 a 决定)
//a=1 发送应答  a=0 不发送应答 准备结束
void Ack_I2c(bit a)
{
    SCL=0;
    if(a==1)SDA=0;        //在此发出应答或非应答信号
    else SDA=1;
    delay4us(2);
    SCL=1;
    delay4us(2);
    SCL=0;                //清时钟线, 钳住 I2C 总线以便继续接收
}

```

8.4 I2C.H

```

#ifndef I2C_H
#define I2C_H
#include "sys.h"

void Start_I2c();        //启动总线
void Stop_I2c();         //停止总线
void SendByte(u8 c);     //字节数据发送函数
u8 RcvByte();            //字节数据接收函数
void Ack_I2c(bit a);

u8 I2c_wait_ack(void);   //等待应答信号到来

#endif

```

9 HX711

9.1 HX711.C

```

#include "HX711.h"

```

```

uint GapValue = 182; //重量系数

/*****
//延时函数
*****/
void Delay__hx711_us(void)
{
    _nop_();_nop_();_nop_();_nop_();
    _nop_();_nop_();_nop_();_nop_();
    _nop_();_nop_();_nop_();_nop_();
    _nop_();_nop_();_nop_();_nop_();
}

/*****
//读取 HX711 重量
*****/
int HX711_Read(void) //增益 128
{
    unsigned long count;
    unsigned char i;
    HX711_DOUT=1;
    Delay__hx711_us();
    HX711_SCK=0;
    count=0;
    EA = 1;
    while(HX711_DOUT);
    EA = 0;
    for(i=0;i<24;i++)
    {
        HX711_SCK=1;
        count=count<<1;
        HX711_SCK=0;
        if(HX711_DOUT)
            count++;
    }
    HX711_SCK=1;
    count=count^0x800000; //第 25 个脉冲下降沿来时，转换数据
    Delay__hx711_us();
    HX711_SCK=0;
    return (count / 2 /GapValue) ; //重量与读数比值为 400 左右
}

```

9.2 HX711.H

```

#ifndef __HX711_H__
#define __HX711_H__

#include "stc15f2k60s2.h" //包含头文件

```

```

#include <intrins.h>

#ifndef uchar
#define uchar unsigned char
#endif

#ifndef uint
#define uint unsigned int
#endif

//IO 设置
sbit HX711_DOUT=P1^4;
sbit HX711_SCK =P1^3;

//函数或者变量声明

extern uint GapValue ; //重量系数

extern void Delay__hx711_us(void);
extern int HX711_Read(void);

#endif

```

10 KEY

10.1 KEY.c

```

#include "key.h"
#include "sys.h"

u8 KeyCode=0;    //给用户使用的键码，17~32 有效
u8 IO_KeyState=0;
u8 IO_KeyState1=0;
u8 IO_KeyHoldCnt=0;    //行列键盘变量
u8 code T_KeyTable[16] = {0,1,2,0,3,0,0,0,4,0,0,0,0,0,0,0};
/*****
行列键扫描程序
使用 XY 查找 4x4 键的方法，只能单键，速度快
*****/



| Y        | P74      | P75      | P76      | P77      |
|----------|----------|----------|----------|----------|
|          |          |          |          |          |
| X        |          |          |          |          |
| P70 ---- | K00 ---- | K01 ---- | K02 ---- | K03 ---- |
|          |          |          |          |          |
| P71 ---- | K04 ---- | K05 ---- | K06 ---- | K07 ---- |
|          |          |          |          |          |
| P72 ---- | K08 ---- | K09 ---- | K10 ---- | K11 ---- |
|          |          |          |          |          |
| P73 ---- | K12 ---- | K13 ---- | K14 ---- | K15 ---- |


```

```

|      |      |      |
*****/
void IO_KeyDelay(void)
{
    u8 i;
    i = 60;
    while(--i)    ;
}

void IO_KeyScan(void)
{
    u8 j;

    j = IO_KeyState1;  //保存上一次状态

    P7 = 0xf0;    //X 低, 读 Y
    IO_KeyDelay();
    IO_KeyState1 = P7 & 0xf0;

    P7 = 0x0f;    //Y 低, 读 X
    IO_KeyDelay();

    IO_KeyState1 |= (P7 & 0x0f);

    IO_KeyState1 ^= 0xff;  //取反

    if(j == IO_KeyState1)  //连续两次读相等
    {
        j = IO_KeyState;
        IO_KeyState = IO_KeyState1;
        if(IO_KeyState != 0)  //有键按下
        {
            F0 = 0;
            if(j == 0)    F0 = 1;  //第一次按下
            else if(j == IO_KeyState)
            {
                if(++IO_KeyHoldCnt >= 20)  //1 秒后重键
                {
                    IO_KeyHoldCnt = 18;
                    F0 = 1;
                }
            }
            if(F0)
            {
                j = T_KeyTable[IO_KeyState >> 4];
                if((j != 0) && (T_KeyTable[IO_KeyState & 0x0f] != 0))
                {
                    KeyCode = (u8)(((j - 1) * 4) + (T_KeyTable[IO_KeyState & 0x0f]) ); //计算键码, 17~32 + 16
                }
            }
        }
    }
}

```



```

        }
    }
    else IO_KeyHoldCnt = 0;
}
P7 = 0xff;
}

```

10.2 KEY.H

```

#ifndef __KEY_H
#define __KEY_H

#include "stc15f2k60s2.h"

void IO_KeyDelay(void);
void IO_KeyScan(void);

#endif

```

11 LCD12864

11.1 LCD12864.C

```

#include"lcd12864.h"

extern uchar code CharCodeLine1[];    //第一行显示字符
extern uchar code CharCodeLine2[];    //第二行显示字符

void LCD12864_Roll(void){ //滚动显示两行字
    uchar i;
    //为显示上半屏第一行字符做准备，地址 0xa0
    //详细参考文章：https://wenku.baidu.com/view/375ec764cc22bcd127ff0c9a.html
    LCD12864_WriteCmd(0xa0);
    while(CharCodeLine1[i]!='\0'){
        LCD12864_WriteData(CharCodeLine1[i]);
        i++;
    }
    i=0;
    //为显示上半屏第二行字符做准备，地址 0xb0
    LCD12864_WriteCmd(0xb0);
    while(CharCodeLine2[i]!='\0')
    {
        LCD12864_WriteData(CharCodeLine2[i]);
        i++;
    }

    for(i=0;i<33;i++)    //上半屏滚动显示
    {
        LCD12864_VerticalRoll(i);
        Delay100ms();    //每行高 16 个像素，两行 32 像素，0.1 秒滚动 1 像素，两行字显示结束共需要 3.2 秒
    }
    LCD12864_WriteCmd(0x30); //恢复基本指令集
}

```

```
/******
```

```
* 函数名      : LCD12864_Delay1ms
```

```
* 函数功能    : 延时 1MS
```

```
* 输入       : c
```

```
* 输出       : 无
```

```
*****/
```

```
void LCD12864_Delay1ms(uint c)
```

```
{
    uchar a,b;
    for(; c>0; c--)
    {
        for(b=199; b>0; b--)
        {
            for(a=1; a>0; a--);
        }
    }
}
```

```
/******
```

```
* 函数名      : LCD12864_WriteCmd
```

```
* 函数功能    : 写命令
```

```
* 输入       : cmd
```

```
* 输出       : 无
```

```
*****/
```

```
void LCD12864_WriteCmd(uchar cmd)
```

```
{
    uchar i;
    i = 0;

    LCD12864_Delay1ms(20); //等待忙完

    LCD12864_RS = 0;      //选择命令
    LCD12864_RW = 0;      //选择写入
    LCD12864_EN = 0;      //初始化使能端

    LCD12864_DATAPORT = cmd; //放置数据

    LCD12864_EN = 1;      //写时序
    LCD12864_Delay1ms(5);
    LCD12864_EN = 0;
}
```

```
/******
```

```
* 函数名      : LCD12864_WriteData
```

```
* 函数功能    : 写数据
```

```

* 输    入        : dat
* 输    出        : 无
*****/

void LCD12864_WriteData(uchar dat)
{
    uchar i;
    i = 0;

    LCD12864_Delay1ms(20); //等待忙完

    LCD12864_RS = 1;      //选择数据
    LCD12864_RW = 0;      //选择写入
    LCD12864_EN = 0;      //初始化使能端

    LCD12864_DATAPORT = dat; //放置数据

    LCD12864_EN = 1;      //写时序
    LCD12864_Delay1ms(5);
    LCD12864_EN = 0;
}

/*****
* 函 数 名        : LCD12864_ReadData
* 函数功能        : 读取数据
* 输    入        : 无
* 输    出        : 读取到的 8 位数据
*****/

uchar LCD12864_ReadData(void)
{
    uchar i, readValue;

    i = 0;

    LCD12864_Delay1ms(20); //等待忙完

    LCD12864_RS = 1;      //选择命令
    LCD12864_RW = 1;
    LCD12864_EN = 0;
    LCD12864_Delay1ms(1); //等待

    LCD12864_EN = 1;
    LCD12864_Delay1ms(1);
    readValue = LCD12864_DATAPORT;
    LCD12864_EN = 0;

    return readValue;
}

```

```

/*****
* 函 数 名      : LCD12864_Init
* 函数功能      : 初始化 LCD12864
* 输 入        : 无
* 输 出        : 无
*****/

void LCD12864_Init()
{

    LCD12864_WriteCmd(0x30); //选择基本指令操作
    LCD12864_WriteCmd(0x0c); //显示开, 关光标
    LCD12864_WriteCmd(0x01); //清除 LCD12864 的显示内容
}

/*****
* 函 数 名      : LCD12864_ClearScreen
* 函数功能      : 在画图模式下, LCD12864 的 01H 命令不能清屏, 所以要自己写一个清
*                * 屏函数
* 输 入        : 无
* 输 出        : 无
*****/

void LCD12864_ClearScreen(void)
{
    uchar i,j;

    LCD12864_WriteCmd(0x34); //开启拓展指令集

    for(i=0;i<32;i++) //因为 LCD 有纵坐标 32 格所以写三十二次
    {
        LCD12864_WriteCmd(0x80+i); //先写入纵坐标 Y 的值
        LCD12864_WriteCmd(0x80); //再写入横坐标 X 的值
        for(j=0;j<32;j++) //横坐标有 16 位, 每位写入两个字节的的数据, 也
        { //就写入 32 次以为当写入两个字节之后横坐标会自
            LCD12864_WriteData(0x00); //动加 1, 所以就不用再次写入地址了。
        }
    }

    LCD12864_WriteCmd(0x36); //0x36 扩展指令里面打开绘图显示
    LCD12864_WriteCmd(0x30); //恢复基本指令集
}

/*****
* 函 数 名      : LCD12864_SetWindow
* 函数功能      : 设置在基本指令模式下设置显示坐标。注意: x 是设置行, y 是设置列
* 输 入        : x, y
* 输 出        : 无
*****/

```

```
*****/
```

```
void LCD12864_SetWindow(uchar x, uchar y)
```

```
{
    uchar pos;

    if(x == 0)        // 第一行的地址是 80H
    {
        x = 0x80;
    }

    else if(x == 1)   //第二行的地址是 90H
    {
        x = 0x90;
    }

    else if(x == 2)   //第三行的地址是 88H
    {
        x = 0x88;
    }

    else if(x == 3)
    {
        x = 0x98;
    }

    pos = x + y;
    LCD12864_WriteCmd(pos);
}
```

```
/*-----*/
```

函数原型: void LCD12864_VericalRoll(uchar N)

函数功能: 将 DDRAM 内容垂直滚动 N 个像素的距离

入口参数: uchar N_Pixel: 卷动的距离大小 (单位为像素), 范围为 0~33 (实际上可以是 0~63, 但 33~63 实际意义不大)

出口参数: 无

返回参数: 无

注意事项: 将 N_Pixel 设为 33 可将 DDRAM 地址 0xa0~0xbf 的内容完全切换到屏幕上, N 为 0 则显示 DDRAM 地址 0x80~0x9f 的内容 (复位默认状态)

```
-----*/
```

```
void LCD12864_VericalRoll(uchar N_Pixel)
```

```
{
    LCD12864_WriteCmd(0x34); //允许绘图模式 (开启扩展指令集模式)
    LCD12864_WriteCmd(0x03); //允许输入滚动位址
    LCD12864_WriteCmd(0x40|N_Pixel); //上卷 N 行 (像素)
}
```

```
*****
```

** 函数功能 : 向液晶连续写入一段字符串

```

** 函数说明：字符可以是任何字符，包括汉字，但是汉字必须是写在一个连续的 16*16 的点阵中
** 函数举例：Write_12864_String("LCD12864 液晶实验")，这段字符串有 8 个英文字符，总共占 4 个 16*16 的点阵，后面的四个同样占 4 个 16*16 的点阵
** 错误举例：Write_12864_String("LCD 液晶显示")，前面的三个字符占了一个半的 16*16 单元的点阵，会导致后面的汉字没法正常显示
** 入口参数：待写入的字符串
** 出口参数：无
*****/
void print(uchar *str)//写入字符串或者汉字
{
    uchar *p;
    p = str;
    while(*p != 0)
    {
        LCD12864_WriteData(*p);
        p = ++str;
    }
}

//定位输出数字
//x: 0 - 3 (行)
//y: 0 - 15 (列)
//num: 0 - 65535 要显示的数字
//num_bit: 0 - 5 数字的位数
void print_n(unsigned char x, unsigned char y, unsigned int num, unsigned char num_bit)
{
    char i;
    unsigned char ii;
    unsigned char dat[6];
    LCD12864_SetWindow(x,y);
    for(i = 0; i < 6; i++) dat[i] = 0; i = 0; //初始化数据
    while(num / 10) //拆位
    {
        dat[i] = num % 10; //最低位
        num /= 10; i++;
    }
    dat[i] = num; //最高位
    ii = i; //保存 dat 的位数
    for(; i >= 0; i--) dat[i] += 48; //转化成 ASCII
    for(i = 0; i < num_bit; i++)
        LCD12864_WriteData(0x20); //清显示区域

    LCD12864_SetWindow(x,y);
    for(i = ii; i >= 0; i--)
        LCD12864_WriteData(dat[i]); //输出数值
}

```

11.2 LCD12864.H

```

#ifndef __LCD12864_H
#define __LCD12864_H

```

```

//---包含头文件---//
#include "stc15f2k60s2.h"
#include "sys.h"

//---重定义关键词---//
#ifndef uchar
#define uchar unsigned char
#endif

#ifndef uint
#define uint unsigned int
#endif

//---定义使用的 IO 口---//
#define LCD12864_DATAPORT P0      //数据 IO 口

sbit LCD12864_RS = P2^0;          //（数据命令）寄存器选择输入    TFT_D10  A8
sbit LCD12864_RW = P2^1;          //液晶读/写控制                TFT_D8   A9
sbit LCD12864_EN = P2^2;          //液晶使能控制                TFT_D9   A10
//---声明全局函数---//
void LCD12864_Delay1ms(uint c);
void LCD12864_WriteCmd(uchar cmd);
void LCD12864_WriteData(uchar dat);
void LCD12864_Init();
void LCD12864_ClearScreen(void);
void LCD12864_SetWindow(uchar x, uchar y);
void LCD12864_VerticalRoll(uchar N_Pixel); //滚动显示
void LCD12864_Roll(void);
void LCD12864_pixel(unsigned char x, unsigned char y, unsigned char attr);
void LCD_line_h(unsigned char y, unsigned char attr);
void print(uchar *str);
void print_n(unsigned char x, unsigned char y,unsigned int num, unsigned char num_bit);

#endif

```

汉字显示坐标

| | | | | | | | | |
|-------|------|-----|-----|-----|-----|-----|-----|-----|
| | X 坐标 | | | | | | | |
| Line1 | 80H | 81H | 82H | 83H | 84H | 85H | 86H | 87H |
| Line2 | 90H | 91H | 92H | 93H | 94H | 95H | 96H | 97H |
| Line3 | 88H | 89H | 8AH | 8BH | 8CH | 8DH | 8EH | 8FH |
| Line4 | 98H | 99H | 9AH | 9BH | 9CH | 9DH | 9EH | 9FH |

3、字符表

| | | | | | | | | | | | | | | | |
|---|---|---|----|----|---|---|---|---|---|---|---|---|---|---|---|
| ☒ | ☒ | ☒ | ♥ | ♠ | ♣ | ♣ | ♣ | ♣ | ♣ | ♣ | ♣ | ♣ | ♣ | ♣ | ♣ |
| ▶ | ◀ | ↑ | !! | ¶ | § | — | ‡ | ↑ | ↓ | → | ← | └ | ↔ | ▲ | ▼ |
| | ! | " | # | \$ | % | & | ' | (|) | * | + | , | - | . | / |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | |
| P | Q | R | S | T | U | V | W | X | Y | Z | [| \ |] | ^ | _ |
| ` | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| p | q | r | s | t | u | v | w | x | y | z | { | | } | ~ | △ |

代码

(02H---7FH)

ST7920 GB 中文字型碼表

| | | | | | | | | | | | | | | | |
|------|------|------|------|------|------|------|------|------|------|------|-----|-----|-----|-----|------|
| A1A0 | 、 | 。 | • | ˘ | ˙ | ˚ | ˛ | ˜ | ˜ | ˜ | ˜ | ˜ | ˜ | ˜ | ˜ |
| A1B0 | “ | ” | (|) | < | > | 《 | 》 | 「 | 」 | 『 | 』 | 【 | 】 | |
| A1C0 | ± | × | ÷ | : | ∧ | ∨ | Σ | Π | U | ∩ | ∈ | :: | √ | ⊥ | // |
| A1D0 | ∩ | ⊙ | ∫ | ∫ | ≡ | ≈ | ≈ | ≈ | ≈ | ≈ | ≈ | ≈ | ≈ | ≈ | ≈ |
| A1E0 | ∴ | ∴ | ∴ | ∴ | ∴ | ∴ | ∴ | ∴ | ∴ | ∴ | ∴ | ∴ | ∴ | ∴ | ∴ |
| A1F0 | ○ | ● | ◎ | ◇ | ◆ | □ | ■ | △ | ▲ | ※ | → | ← | ↑ | ↓ | ＝ |
| A2A0 | | | | | | | | | | | | | | | |
| A2B0 | 1. | 2. | 3. | 4. | 5. | 6. | 7. | 8. | 9. | 10. | 11. | 12. | 13. | 14. | 15. |
| A2C0 | 16. | 17. | 18. | 19. | 20. | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | (10) |
| A2D0 | (11) | (12) | (13) | (14) | (15) | (16) | (17) | (18) | (19) | (20) | ① | ② | ③ | ④ | ⑤ |
| A2E0 | ⑥ | ⑦ | ⑧ | ⑨ | ⑩ | (一) | (二) | (三) | (四) | (五) | (六) | (七) | (八) | (九) | (十) |
| A2F0 | I | II | III | IV | V | VI | VII | VIII | IX | X | XI | XII | | | |
| A3A0 | ! | " | # | ¥ | % | & | ' | (|) | * | + | - | . | / | |
| A3B0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > |
| B0D0 | 靶 | 把 | 耙 | 坝 | 霸 | 罢 | 爸 | 白 | 柏 | 百 | 摆 | 佰 | 败 | 拜 | 裨 |
| B0E0 | 斑 | 搬 | 扳 | 般 | 颁 | 扳 | 版 | 扮 | 拌 | 伴 | 瓣 | 办 | 办 | 办 | 办 |
| B0F0 | 梆 | 榜 | 膀 | 绑 | 棒 | 磅 | 蚌 | 镑 | 傍 | 谤 | 苞 | 胞 | 包 | 褒 | 剥 |
| B1A0 | 薄 | 雹 | 保 | 堡 | 饱 | 宝 | 抱 | 报 | 暴 | 豹 | 鲍 | 爆 | 杯 | 碑 | 悲 |
| B1B0 | 卑 | 北 | 辈 | 背 | 贝 | 倍 | 狈 | 备 | 惫 | 焙 | 被 | 奔 | 笨 | 本 | 笨 |
| B1C0 | 崩 | 绷 | 甬 | 泵 | 蹦 | 迸 | 鼻 | 比 | 鄙 | 笔 | 彼 | 碧 | 蓖 | 蔽 | 毕 |
| B1D0 | 毙 | 砵 | 币 | 庇 | 痹 | 闭 | 蔽 | 弊 | 必 | 辟 | 臂 | 避 | 陛 | 鞭 | 边 |
| B1E0 | 编 | 贬 | 扁 | 便 | 变 | 卞 | 辨 | 辩 | 辨 | 遍 | 标 | 彪 | 膘 | 表 | 鳖 |
| B1F0 | 别 | 瘳 | 彬 | 斌 | 濒 | 滨 | 宾 | 宾 | 兵 | 冰 | 柄 | 丙 | 秉 | 饼 | 炳 |
| B2A0 | 病 | 并 | 玻 | 波 | 播 | 拨 | 钵 | 波 | 博 | 勃 | 搏 | 铂 | 箔 | 伯 | 帛 |
| B2B0 | 舶 | 脖 | 膊 | 泊 | 驳 | 捕 | 卜 | 哺 | 补 | 埠 | 不 | 布 | 步 | 簿 | 部 |
| B2C0 | 怖 | 擦 | 猜 | 裁 | 材 | 才 | 财 | 睬 | 踩 | 采 | 彩 | 菜 | 蔡 | 餐 | 参 |
| B2D0 | 残 | 惭 | 惨 | 灿 | 苍 | 舱 | 仓 | 沧 | 藏 | 操 | 糙 | 槽 | 曹 | 草 | 策 |
| B2E0 | 侧 | 册 | 测 | 层 | 蹭 | 插 | 叉 | 茬 | 茶 | 查 | 楂 | 搽 | 察 | 岔 | 诧 |
| B2F0 | 拆 | 柴 | 豺 | 搀 | 惨 | 蝉 | 馋 | 谗 | 缠 | 铲 | 产 | 阐 | 颤 | 昌 | 猖 |
| B3A0 | 场 | 尝 | 常 | 长 | 偿 | 肠 | 厂 | 敞 | 畅 | 唱 | 倡 | 超 | 抄 | 钞 | 朝 |
| B3B0 | 嘲 | 潮 | 巢 | 吵 | 炒 | 车 | 扯 | 撤 | 掣 | 彻 | 澈 | 郴 | 臣 | 辰 | 尘 |

12 MOTOR

12.1 MOTOR.C

```
#include "motor.h"
#include "sys.h"

//      D C B A
//  7 6 5 4 3 2 1 0    P1口
//  0 0 0 1 0 0 0 0

uchar phasecw[4] = {0x10,0x08,0x04,0x02}; //正转 电机导通相序 D-C-B-A
```



```
uchar phaseccw[4]={0x02,0x04,0x08,0x10}; //反转 电机导通相序 A-B-C-D
```

```
uchar speed;
```

```
//逆时针转动
```

```
void MotorCCW(void)
```

```
{
    uchar i;
    for(i=0;i<4;i++)
    {
        MotorData=phaseccw[i];
        delay_ms(speed); //转速调节
    }
}
```

```
//顺时针转动
```

```
void MotorCW(void)
```

```
{
    uchar 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;
}
```

```
//针对 STC15W4K56S4 系列 IO 口初始化
```

```
//io 口初始化 P0 P1 P2 P3 P4 为准双向 IO 口
```

```
//注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
```

```
//      高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
```

```
//相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
```

```
//      P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
```

```
void motor_init(){
```

```
    //设置 STC 单片机的 P15 P16 为推挽输出
    P1M1 &= ~(1<<5),P1M0 |= (1<<5);
    P1M1 &= ~(1<<6),P1M0 |= (1<<6);
}
```

```
void dc_motor_stop(){
```

```
    moto1=0;
    moto2=0;
}
```

```

void dc_motor_run(){
    moto1=1;
    moto2=1;
}

```

12.2 MOTOR.H

```

#ifndef MOTOR_H
#define MOTOR_H

#include "stc15f2k60s2.h"

//---重定义关键词---//
#ifndef uchar
#define uchar unsigned char
#endif

#ifndef uint
#define uint unsigned int
#endif

#ifndef ushort
#define ushort unsigned short
#endif

sbit moto1=P1^5;
sbit moto2=P1^6;

#define MotorData P1
//步进电机控制接口定义

void motor_init();
void dc_motor_stop();
void dc_motor_run();
void MotorCCW(void);
void MotorCW(void);
void MotorStop(void);

#endif

```

13 MPU6050

13.1 MPU6050.C

```

#include "mpu6050.h"
#include <INTRINS.H>

sbit SCL=P6^7; //IIC 时钟引脚定义
sbit SDA=P6^6; //IIC 数据引脚定义

```

```

//*****
//延时 5 微秒(STC90C52RC@12M)
//不同的工作环境,需要调整此函数
//当改用 1T 的 MCU 时,请调整此延时函数
//*****
void Delay5us()
{
    _nop_();_nop_();_nop_();_nop_();
    _nop_();_nop_();_nop_();_nop_();
    _nop_();_nop_();_nop_();_nop_();
    _nop_();_nop_();_nop_();_nop_();
    _nop_();_nop_();_nop_();_nop_();
    _nop_();_nop_();_nop_();_nop_();
}
//*****
//I2C 起始信号
//*****
void I2C_Start()
{
    SDA = 1;           //拉高数据线
    SCL = 1;           //拉高时钟线
    Delay5us();         //延时
    SDA = 0;           //产生下降沿
    Delay5us();         //延时
    SCL = 0;           //拉低时钟线
}
//*****
//I2C 停止信号
//*****
void I2C_Stop()
{
    SDA = 0;           //拉低数据线
    SCL = 1;           //拉高时钟线
    Delay5us();         //延时
    SDA = 1;           //产生上升沿
    Delay5us();         //延时
}
//*****
//I2C 发送应答信号
//入口参数:ack (0:ACK 1:NAK)
//*****
void I2C_SendACK(bit ack)
{
    SDA = ack;         //写应答信号
    SCL = 1;           //拉高时钟线
    Delay5us();         //延时
    SCL = 0;           //拉低时钟线
}

```

```

    Delay5us();                //延时
}

//*****
//I2C 接收应答信号
//*****
bit I2C_RecvACK()
{
    SCL = 1;                    //拉高时钟线
    Delay5us();                //延时
    CY = SDA;                  //读应答信号
    SCL = 0;                    //拉低时钟线
    Delay5us();                //延时
    return CY;
}

//*****
//向 I2C 总线发送一个字节数据
//*****
void I2C_SendByte(uchar dat)
{
    uchar 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
        Delay5us();            //延时
        SCL = 0;                //拉低时钟线
        Delay5us();            //延时
    }
    I2C_RecvACK();
}

//*****
//从 I2C 总线接收一个字节数据
//*****
uchar I2C_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
        Delay5us();            //延时
        dat |= SDA;             //读数据
        SCL = 0;                //拉低时钟线
        Delay5us();            //延时
    }
    return dat;
}

```

```

}

//*****
//向 I2C 设备写入一个字节数据
//*****
void Single_WriteI2C(uchar REG_Address,uchar REG_data)
{
    I2C_Start();           //起始信号
    I2C_SendByte(SlaveAddress); //发送设备地址+写信号
    I2C_SendByte(REG_Address); //内部寄存器地址,
    I2C_SendByte(REG_data);    //内部寄存器数据,
    I2C_Stop();              //发送停止信号
}

//*****
//从 I2C 设备读取一个字节数据
//*****
uchar Single_ReadI2C(uchar REG_Address)
{
    uchar REG_data;
    I2C_Start();           //起始信号
    I2C_SendByte(SlaveAddress); //发送设备地址+写信号
    I2C_SendByte(REG_Address); //发送存储单元地址, 从 0 开始
    I2C_Start();           //起始信号
    I2C_SendByte(SlaveAddress+1); //发送设备地址+读信号
    REG_data=I2C_RecvByte(); //读出寄存器数据
    I2C_SendACK(1);         //接收应答信号
    I2C_Stop();             //停止信号
    return REG_data;
}

//*****
//初始化 MPU6050
//*****
void InitMPU6050()
{
    Single_WriteI2C(PWR_MGMT_1, 0x00); //解除休眠状态
    Single_WriteI2C(SMPLRT_DIV, 0x07);
    Single_WriteI2C(CONFIG, 0x06);
    Single_WriteI2C(GYRO_CONFIG, 0x18);
    Single_WriteI2C(ACCEL_CONFIG, 0x01);
}

//*****
//合成数据
//*****
int GetData(uchar REG_Address)
{
    char H,L;
    H=Single_ReadI2C(REG_Address);
    L=Single_ReadI2C(REG_Address+1);
    return (H<<8)+L; //合成数据
}

```

13.2 MPU6050.H

```
#ifndef __MPU6050_H__
#define __MPU6050_H__

#include "stc15f2k60s2.h"

//*****
// 定义 MPU6050 内部地址
//*****

#define SMPLRT_DIV      0x19      //陀螺仪采样率，典型值：0x07(125Hz)
#define CONFIG          0x1A      //低通滤波频率，典型值：0x06(5Hz)
#define GYRO_CONFIG     0x1B      //陀螺仪自检及测量范围，典型值：0x18(不自检，2000deg/s)
#define ACCEL_CONFIG    0x1C      //加速计自检、测量范围及高通滤波频率，典型值：0x01(不自检，2G，5Hz)
#define ACCEL_XOUT_H    0x3B
#define ACCEL_XOUT_L    0x3C
#define ACCEL_YOUT_H    0x3D
#define ACCEL_YOUT_L    0x3E
#define ACCEL_ZOUT_H    0x3F
#define ACCEL_ZOUT_L    0x40
#define TEMP_OUT_H      0x41
#define TEMP_OUT_L      0x42
#define GYRO_XOUT_H     0x43
#define GYRO_XOUT_L     0x44
#define GYRO_YOUT_H     0x45
#define GYRO_YOUT_L     0x46
#define GYRO_ZOUT_H     0x47
#define GYRO_ZOUT_L     0x48
#define PWR_MGMT_1      0x6B      //电源管理，典型值：0x00(正常启用)
#define WHO_AM_I        0x75      //IIC 地址寄存器(默认数值 0x68，只读)
#define SlaveAddress    0xD0      //IIC 写入时的地址字节数据，+1 为读取

//---重定义关键词---//
#ifndef uchar
#define uchar unsigned char
#endif

#ifndef uint
#define uint unsigned int
#endif

#ifndef ushort
#define ushort unsigned short
#endif

void InitMPU6050(); //初始化
MPU6050
void Delay5us();
```

```

void I2C_Start();
void I2C_Stop();
void I2C_SendACK(bit ack);
bit I2C_RecvACK();
void I2C_SendByte(uchar dat);
uchar I2C_RecvByte();
void I2C_ReadPage();
void I2C_WritePage();
void display_ACCEL_x();
void display_ACCEL_y();
void display_ACCEL_z();
uchar Single_ReadI2C(uchar REG_Address);           //读取 I2C 数据
void Single_WriteI2C(uchar REG_Address,uchar REG_data); //向 I2C 写入数据
int GetData(uchar REG_Address);

#endif

```

14 TIMER

14.1 TIMER.C

```

#include "sys.h"
#include "timer.h"
#include "stc15f2k60s2.h"
#include "ad.h"

//定制器和步进电机控制代码

u16 TimerCount = 0;
u8 PulseNum = 0;    //脉冲计数
u8 MoterCount = 0;  //步进电机控制计数
u16 AdcCount = 0;   //adc 采样计数
u16 WaiteTimeCount = 0; //静止状态等待时间计数

extern u8 MoterSpeed; //定制器 2 控制步进电机转速
extern u8 AdcFlag ;   //adc 采样标志
extern u16 ad;         //ADC 采样值

// 步进电机逆时针旋转相序表
u8 code CCW[8]={0x08,0x0c,0x04,0x06,0x02,0x03,0x01,0x09};

// 步进电机四相的引脚定义
sbit A1 = P1^1;
sbit B1 = P1^2;
sbit C1 = P1^3;

```

```

sbit D1 = P1^4;

// 给步进电机每个相赋值
void Give(unsigned char dat)
{
    A1 = dat & 0x08;
    B1 = dat & 0x04;
    C1 = dat & 0x02;
    D1 = dat & 0x01;
}

//定时器0 初始化
void Timer0Init(void)//10 毫秒@24.000MHz
{
    AUXR &= 0x7F; //定时器时钟 12T 模式
    TMOD &= 0xF0; //设置定时器模式
    TL0 = 0xE0;      //设置定时初值
    TH0 = 0xB1;      //设置定时初值
    TF0 = 0;         //清除 TF0 标志
    TR0 = 1;         //定时器0 开始计时
}

//定时器2 初始化
void Timer2Init(void)//250 微秒@24.000MHz
{
    AUXR &= 0xFB; //定时器时钟 12T 模式
    T2L = 0x0C;      //设置定时初值
    T2H = 0xFE;      //设置定时初值
    AUXR |= 0x10; //定时器2 开始计时
}

void tm0() interrupt 1 using 1 //定时器0 中断
{
    WaiteTimeCount ++ ;      //10 毫秒加一
}

void tm2() interrupt 12 //定时器2 中断
{
    #if DRIVING_SIMULATION

    TimerCount ++ ;      //定时器2 脉冲模拟效果
    #endif
}

```



```

    if(TimerCount>4000) //定时超过 1 秒
    {
        PulseNum ++ ;
        TimerCount = 0;
    }

#else

    TimerCount ++ ;
    if(TimerCount >MoterSpeed) //定时超过 MoterSpeed 为步进电机速度控制
    {
        MoterCount ++ ;
        TimerCount = 0;
        Give(CCW[MoterCount%8]); //控制电机
    }
    if( MoterCount%64==63 ) //定时 ADC 采样标志位
    {
        MoterCount = 0;
        AdcFlag = 1;
    }

#endif

}

```

14.2 TIMER.H

```

#ifndef __TIMER_H
#define __TIMER_H

void Give(unsigned char dat); //步进电机驱动

void Timer2Init(void);
void Timer0Init(void);
void tm2();
void tm0();
#endif

```