一些常用的库

```
#ifndef  _MAIN_H_
#define  _MAIN_H_

#include "STC12C5A60S2.H"        //芯片头文件
#include"intrins.h"
#include "int_it.h"                  // 外部中断
#include "time.h"                                    //定时中断
#include "usart.h"                                   //串口
#include "ADC.h"                                     //片内 ADC
#include "PWM.h"                                     //PWM 输出
#include "HX711.h"         //压力传感器
#include "Digital_tube.h" // 数码管
#include "AT24C02.h"          //片外 EEPROM
#include "DS18B20.h"             // 温度传感器
#include "ds1302.h"              //时钟模块
#include "IO_expand.h"           //74HC595/138
#include "LCD1602.h"                    //LCD1602
#include "parallel_12864.h" // 并行 LCD12864
#include "PCF8591.h"                    // 片外 AD/DA 模块
#include "stepper_motor.h"// 4 步进电机
#include "delay.h"         //延时函数
#include "buzz.h"          //蜂鸣器
#include "key.h"           //按键
#include "as608.h"         //指纹
#include "mpu6050.h"       //6 轴姿态传感器
#endif

#include "STC12C5A60S2.H"
#include "main.h"

void main(){
        while(1){
        }
}
```

1.adc

adc.h

```
/*********************************************************************************
***********************
** 文件功能 : STC12C5A60S2 内置 AD 驱动程序
** 工程作者 : Blue Sky Teams——ZZL
** 工程版本 : V1.0
*********************************************************************************
**********************/
#ifndef _ADC_H_
#define _ADC_H_

  #ifndef uchar
        #define uchar unsigned char
```

```
  #endif

  #ifndef uint
        #define uint  unsigned int
  #endif

#include"intrins.h"
#include "STC12C5A60S2.H"

/*******************************用于配置P1口对应管脚为AD模拟输入口*****************
*********************/
#define ADC_PORT0                       0X01
#define ADC_PORT1                       0X02
#define ADC_PORT2                       0X04
#define ADC_PORT3                       0X08
#define ADC_PORT4                       0X10
#define ADC_PORT5                       0X20
#define ADC_PORT6                       0X40
#define ADC_PORT7                       0X80
#define ADC_PORTALL                     0XFF

/*********************************用于获取对应通道的电压值**********************
*******************/
#define ADC_CH0                         0X00
#define ADC_CH1                         0X01
#define ADC_CH2                         0X02
#define ADC_CH3                         0X03
#define ADC_CH4                         0X04
#define ADC_CH5                         0X05
#define ADC_CH6                         0X06
#define ADC_CH7                         0X07

/*****************************************定义AD转换速度***********************
**********************/
#define ADC_SPEEDLL_540  0X00
#define ADC_SPEEDLL_360         0X20
#define ADC_SPEEDLL_180         0X40
#define ADC_SPEEDLL_90          0X60

/*************************************定义转换控制寄存器控制位*****************
*******************/
#define ADC_POWER               0X80            //电源控制位
#define ADC_FLAG                0X10            //转换结束标志位
#define ADC_START               0X08            //转换开始位


/*****************************************************************************
***********************
** 函数功能 : 内置ADC的初始化配置
** 函数说明 : 使用内置ADC时需要先配置对应的P1口的管脚为模拟输入
** 入口参数 : port: 需要配置为模拟输入的通道，使用或运算可以同时配置多个管脚
**                          如：ADC_Init(ADC_PORT0 | ADC_PORT1 | ADC_PORT2)调用此函数
后可以同时配置P1^0,P1^1,P1^2为模拟输入
** 出口参数 : 无
*****************************************************************************
**********************/
extern void ADC_Init(unsigned char port);


/*****************************************************************************
***********************
** 函数功能 : 获取ADC对应通道的电压值
** 函数说明 : 每次只能获取一个通道的电压值，不同通道需要分别调用该函数获取
** 入口参数 : channel：获取该通道的电压值
** 出口参数 : result：  当前通道的电压值
*****************************************************************************
**********************/
extern float GetADCResult(unsigned char channel); //读取通道ch的电压值

#endif
```

adc.c

```
/*****************************************************************************
***********************
** 文件功能 : STC12C5A60S2内置AD驱动程序
** 工程作者 : Blue Sky Teams——ZZL
** 工程版本 : V1.0
*****************************************************************************
**********************/
#include "ADC.h"

/*****************************************************************************
***********************
** 函数功能 : 内置ADC的初始化配置
** 函数说明 : 使用内置ADC时需要先配置对应的P1口的管脚为模拟输入
** 入口参数 : port: 需要配置为模拟输入的通道，使用或运算可以同时配置多个管脚
**                          如：ADC_Init(ADC_PORT0 | ADC_PORT1 | ADC_PORT2)调用此函数
后可以同时配置P1^0,P1^1,P1^2为模拟输入
** 出口参数 : 无
*****************************************************************************
**********************/
void ADC_Init(unsigned char port)
{
        P1ASF=port;//设置AD转换通道
        ADC_RES=0;//清空转换结果
        ADC_CONTR=ADC_POWER | ADC_SPEEDLL_540;//打开AD转化器电源
//      IE=0XA0;//开启总中断，ADC中断
        _nop_();
        _nop_();
        _nop_();
        _nop_();
}
/*****************************************************************************
***********************
** 函数功能 : 获取ADC对应通道的电压值
```

```
** 函数说明 ： 每次只能获取一个通道的电压值，不同通道需要分别调用该函数获取
** 入口参数 ： channel：获取该通道的电压值
** 出口参数 ： result：  当前通道的电压值
*******************************************************************************
*********************/
float GetADCResult(unsigned char channel)//读取通道ch的电压值
{
        unsigned int ADC_RESULT = 0;//用来存放结果
        float result;
        ADC_CONTR = ADC_POWER | ADC_SPEEDLL_540 | ADC_START | channel;//开始转换，并
设置测量通道为P1^0
        _nop_();//需经过四个CPU时钟延时，上述值才能保证被设进ADC_CONTR控制寄存器
        _nop_();
        _nop_();
        _nop_();
        while(!(ADC_CONTR & ADC_FLAG));//等待转换结束
        ADC_CONTR &= ~ADC_FLAG;//软件清除中断控制位
        ADC_RESULT = ADC_RES;
        ADC_RESULT = (ADC_RESULT << 2) | (0x02 & ADC_RESL);     //默认数据存储方式：
高八位在ADC_RES,低二位在ADC_RESL低二位
        result = ADC_RESULT * 5.0 / 1024.0 ;  //基准电压为电源电压5V，10的分辨率，即
1024
        return result;
}
```

2.at24c02//EEPROM

at24c02.h

```
/*******************************************************************************
**********************
** 文件功能 ： AT24C02 驱动程序
** 文件说明 ： AT24C02的的三位地址线全部接地，故其地址为0xa0
** 工程作者 ： Blue Sky Teams——ZZL
** 工程版本 ： V1.0
*******************************************************************************
*********************/
#ifndef _AT24C02_H_
#define _AT24C02_H_
  #ifndef uchar
        #define uchar unsigned char
  #endif

  #ifndef uint
        #define uint  unsigned int
  #endif
#include"intrins.h"
#include "STC12C5A60S2.H"
sbit AT24C02_SDA=P2^7;//双向数据端口
sbit AT24C02_SCL=P2^6;//串行时钟
/*******************************************************************************
**********************
** 函数功能 ： IIC 起始信号
** 函数说明 ： SCL 线为高电平期间，SDA 线由高电平向低电平的变化表示起始信号
** 入口参数 ： 无
** 出口参数 ： 无
*******************************************************************************
*********************/
extern void AT24C02_Start();

/*******************************************************************************
**********************
** 函数功能 ： IIC 终止信号
** 函数说明 ： SCL 线为高电平期间，SDA 线由低电平向高电平的变化表示终止信号。
** 入口参数 ： 无
** 出口参数 ： 无
*******************************************************************************
*********************/
extern void AT24C02_Stop();

/*******************************************************************************
**********************
** 函数功能 ： IIC 应答信号
** 函数说明 ： 等待应答即SDA为低，若等待一定时间还没应答，默认为接受完了
** 入口参数 ： 无
** 出口参数 ： 无
*******************************************************************************
*********************/
extern void AT24C02_Ack();

/*******************************************************************************
**********************
** 函数功能 ： 发送一个字节
** 入口参数 ： dat：待发送的字节
** 出口参数 ： 无
*******************************************************************************
*********************/
extern void AT24C02_Write_Byte(dat);

/*******************************************************************************
**********************
** 函数功能 ： 接收一个字节
** 入口参数 ： 无
** 出口参数 ： dat：接收到的字节
*******************************************************************************
*********************/
extern uchar AT24C02_Read_Byte();

/*******************************************************************************
**********************
** 函数功能 ： AT24C02的初始化函数
** 函数说明 ： AT24C02初始化时SCL、SDA均为高电平
** 入口参数 ： 无
** 出口参数 ： 无
*******************************************************************************
```

```
**********************/
extern void AT24C02_Init();

/*******************************************************************************
************************
** 函数功能 : 随机地址存储字节
** 函数说明 : 随机写入一个地址，将字节存在那个地址上
** 入口参数 : dat：待存储的字节
**            add：存储字节的地址
** 出口参数 : 无
*******************************************************************************
**********************/
extern void AT24C02_Write_Add(uchar dat,uchar add);

/*******************************************************************************
************************
** 函数功能 : 随机地址读取字节
** 函数说明 : 写入一个地址，读取这个地址上的字节
** 入口参数 :  add：待读取字节的地址
** 出口参数 : 无
*******************************************************************************
**********************/
extern uchar AT24C02_Read_Add(uchar add);

/*******************************************************************************
************************
** 函数功能 : 随机地址读取一页
** 函数说明 : 页写方式，地址必须满足页方式，才能被全部写进EEPROM；
**            如地址要为0x00,0x08，0x10这样写入的8个字节才能全部写进去，否则如写0x01
则写入数据最后一个将无法写进！
** 入口参数 : add：待存储字符串的地址
**            dat：待存储的字符串
** 出口参数 : 无
*******************************************************************************
**********************/
extern void AT24C02_Write_Page(uchar *dat,uchar add);

/************************** (C) COPYRIGHT 2011 Blue Sky Teams *****END OF FILE*****
************************/

#endif

/*
void main()
{
        uchar temp;
        LCD_1602_Init();
        AT24C02_Init();          //I2C总线初始化
        Write_Add('A',0x03);//向AT24C02内部地址为0x03处写入一个字节的数据
        Delay_Ms(100);
        temp = Read_Add(0x03); //从AT24C02内部地址为0x03处读出刚写入的数据
        Write_1602_String("The data is:",0x80);
        Write_1602_Data(temp);
        while(1)
        {

        }
}
*/
```

at24c02.c

```
/*******************************************************************************
************************
** 文件功能 : AT24C02 驱动程序
** 文件说明 : AT24C02 的的三位地址线全部接地，故其地址为0xa0
** 工程作者 : Blue Sky Teams——ZZL
** 工程版本 : V1.0
*******************************************************************************
**********************/
#include"AT24C02.h"
/*******************************************************************************
************************
** 函数功能 : IIC 起始信号
** 函数说明 : SCL 线为高电平期间，SDA 线由高电平向低电平的变化表示起始信号
** 入口参数 : 无
** 出口参数 : 无
*******************************************************************************
**********************/
void AT24C02_Start()
{
        AT24C02_SDA=1;_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_
();
        AT24C02_SCL=1;_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_
();
        AT24C02_SDA=0;_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_
();
}
/*******************************************************************************
************************
** 函数功能 : IIC 终止信号
** 函数说明 : SCL 线为高电平期间，SDA 线由低电平向高电平的变化表示终止信号。
** 入口参数 : 无
** 出口参数 : 无
*******************************************************************************
**********************/
void AT24C02_Stop()
{
        AT24C02_SDA=0;_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_
();
        AT24C02_SCL=1;_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_
();
        AT24C02_SDA=1;_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_
();
}
/*******************************************************************************
```

```
**********************
** 函数功能 : IIC 应答信号
** 函数说明 : 等待应答即 SDA 为低，若等待一定时间还没应答，默认为接受完了
** 入口参数 : 无
** 出口参数 : 无
*******************************************************************************
**********************/
void AT24C02_Ack()
{
        uchar i=0;
        AT24C02_SCL=1;_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_
();
        while((AT24C02_SDA==1)&&(i<250))i++;
        AT24C02_SCL=0;_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_
();
}
/*******************************************************************************
**********************
** 函数功能 : 发送一个字节
** 入口参数 : dat：待发送的字节
** 出口参数 : 无
*******************************************************************************
**********************/
void AT24C02_Write_Byte(dat)
{
        uchar temp,i;
        temp=dat;
        for(i=0;i<8;i++)
        {
                temp=temp<<1;
                AT24C02_SCL=0;_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_
();_nop_();
                AT24C02_SDA=CY;_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop
_();_nop_();
                AT24C02_SCL=1;_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_
();_nop_();
        }
        AT24C02_SCL=0;_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_
();
        AT24C02_SDA=1;_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_
();//释放数据总线，以便后面的应答信号
}
/*******************************************************************************
**********************
** 函数功能 : 接收一个字节
** 入口参数 : 无
** 出口参数 : dat：接收到的字节
*******************************************************************************
**********************/
uchar AT24C02_Read_Byte()
{
        uchar dat,i;
    AT24C02_SCL=0;_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();
        AT24C02_SDA=1;_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_
();//释放数据总线
        for(i=0;i<8;i++)
        {
                AT24C02_SCL=1;_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_
();_nop_();
                dat=(dat<<1)|AT24C02_SDA;_nop_();_nop_();_nop_();_nop_();_nop_();_n
op_();_nop_();_nop_();
                AT24C02_SCL=0;_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_
();_nop_();
        }
        return(dat);
}
/*******************************************************************************
**********************
** 函数功能 : AT24C02 的初始化函数
** 函数说明 : AT24C02 初始化时 SCL、SDA 均为高电平
** 入口参数 : 无
** 出口参数 : 无
*******************************************************************************
**********************/
void AT24C02_Init()
{
        AT24C02_SCL=1;_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_
();
        AT24C02_SDA=1;_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_
();
}
/*******************************************************************************
**********************
** 函数功能 : 随机地址存储字节
** 函数说明 : 随机写入一个地址，将字节存在那个地址上
** 入口参数 : dat：待存储的字节
**            add：存储字节的地址
** 出口参数 : 无
*******************************************************************************
**********************/
void AT24C02_Write_Add(uchar dat,uchar add)
{
        AT24C02_Stop();//终止
        AT24C02_Start();//起始
        AT24C02_Write_Byte(0xa0);//从器件地址，即将进行写
        AT24C02_Ack();//应答
        AT24C02_Write_Byte(add);//字节地址
        AT24C02_Ack();//应答
        AT24C02_Write_Byte(dat);//写的数据
        AT24C02_Ack();//应答
        AT24C02_Stop();//终止
}
/*******************************************************************************
**********************
** 函数功能 : 随机地址读取字节
** 函数说明 : 写入一个地址，读取这个地址上的字节
```

```
** 入口参数 ：  add：待读取字节的地址
** 出口参数 ：无
*******************************************************************************
*********************/
uchar AT24C02_Read_Add(uchar add)
{
        uchar dat;
        AT24C02_Stop();//终止
        AT24C02_Start();//起始
        AT24C02_Write_Byte(0xa0);//从器件地址
        AT24C02_Ack();//应答
        AT24C02_Write_Byte(add);//字节地址
        AT24C02_Ack();//应答
        AT24C02_Start();//起始
        AT24C02_Write_Byte(0xa1);//从器件地址，即将进行读
        AT24C02_Ack();//应答
        dat=AT24C02_Read_Byte();//读回数据
        AT24C02_Stop();//终止
        return(dat);
}
/*******************************************************************************
**********************
** 函数功能 ：随机地址读取一页
** 函数说明 ：页写方式，地址必须满足页方式，才能被全部写进EEPROM;
**             如地址要为0x00,0x08，0x10这样写入的8个字节才能全部写进去，否则如写0x01
则写入数据最后一个将无法写进!
** 入口参数 ：add：待存储字符串的地址
**            dat：待存储的字符串
** 出口参数 ：无
*******************************************************************************
*********************/
void AT24C02_Write_Page(uchar *dat,uchar add)
{
        uchar q;
        AT24C02_Stop();//终止
        AT24C02_Start();
        AT24C02_Write_Byte(0xa0);
        AT24C02_Ack();
        AT24C02_Write_Byte(add);
        AT24C02_Ack();
        for(q=0;dat[q]!='\0';q++)
        {
                AT24C02_Write_Byte(dat[q]);
                AT24C02_Ack();
        }
        AT24C02_Stop();
}
/************************ (C) COPYRIGHT 2011 Blue Sky Teams *****END OF FILE*****
***********************/
```

3.buzzle//蜂鸣器

buzz.h

```
#ifndef _BUZZ_H_
#define _BUZZ_H_
#include<STC12C5A60S2.h>

sbit buzz=P1^0;

void Buzz_Times(unsigned char times);

#endif
```

buzz.c

```
#include "buzz.h"
#include "delay.h"

void Buzz_Times(unsigned char times)
{
        unsigned char i=0;
        for(i=0;i<times;i++)
        {
                buzz=0;
                Delay_Ms(200);
                buzz=1;
                Delay_Ms(200);
        }
}
```

4.delay 延时函数

delay.h

```
#ifndef __DELAY_H__
#define __DELAY_H__

#include<STC12C5A60S2.h>
#include"intrins.h"

#ifndef uchar
        #define uchar unsigned char
#endif

#ifndef uint
        #define uint  unsigned int
#endif

void Delay_Us(uchar n);
void Delay_Ms(uint time);

#endif
```

delay.c

```
#include"delay.h"
/*****************************************************************************
***********************
** 函数功能 : 延时函数
** 函数说明 : 利用软件延时，占用CPU，经调试最小单位大约为1us
** 入口参数 : time:需要延时的时间，单位us
** 出口参数 : 无
*****************************************************************************
**********************/
void Delay_Us(uchar n)
{
    while (n--)
    {
        _nop_();
        _nop_();
    }
}
/*****************************************************************************
***********************
** 函数功能 : 延时函数
** 函数说明 : 利用软件延时，占用CPU，经调试最小单位大约为1ms
** 入口参数 : time:需要延时的时间，单位ms
** 出口参数 : 无
*****************************************************************************
**********************/
void Delay_Ms(uint time)
{
        uint i,j;
        for(i = 0;i < time;i ++)
                for(j = 0;j < 930;j ++);
}

/*****************************************************************************
***********************
函数功能 : 延时函数

不知道多久
*****************************************************************************
/
void Delay(uint time)             //int 型数据为16 位,所以最大值为65535
{
        uint  i,j;                  //定义变量i,j,用于循环语句
        for(i=0;i<time;i++)         //for 循环,循环50*time 次
        for(j=0;j<50;j++);       //for 循环,循环50 次
}
```

5.dth11//温湿度传感器

dth11.h

```
//
// Created by Jay on 2022/6/11.
//
#ifndef __DHT11_H
#define __DHT11_H
#include"intrins.h"
#include "STC12C5A60S2.H"
#include "delay.h"

#ifndef uchar
#define uchar unsigned char
#endif

#ifndef uint
#define uint  unsigned int
#endif

uchar dat_r[4];          //用于存放从DHT11 读取到的数值
sbit DATA=P3^3;   //定义数据线,DHT11 的2 脚，2 脚要上拉5.1K 电阻

#endif //__DHT11_H
```

dth11.c

```
//
// Created by Jay on 2022/6/11.
//
#include "DHT11.h"

/*****************************************************************************
***********
函数名：DHT11 启动函数
调  用：无
参  数：无
返回值：无
结  果：DHT11 开始工作
备  注:
*****************************************************************************
***********/
void DHT11_start()                //主机控制DHT11 开始工作
{
    DATA=1;                        // 主线空闲状态
    _nop_();
    _nop_();
    DATA=0;                        // 主机发送开始工作信号
    Delay_Ms(20);   //延时18ms 以上
    DATA=1;                        // 拉高并延时等待DHT11 响应
    Delay_Us(30);
}
/*****************************************************************************
***********
函数名：DHT11 读数据函数
调  用：? = DHT11_rec_byte();
参  数：无
返回值：
```

```
结  果：读DHT11 数据并保存到dat
备  注：
**************************************************************************************
***********/
uchar DHT11_rec_byte()      //接收一个字节
{
    uchar i,dat=0;
    for(i=0;i<8;i++)          //从高到低依次接收 8 位数据
    {
        while(!DATA);         //等待 50us 低电平过去
        Delay_Us(60);    //延时 60us，如果还为高则数据为 1，否则为 0
        dat<<=1;              //左移位使正确接收 8 位数据，数据为 0 时直接移位
        if(DATA==1)           //数据为 1 时，使 dat 加 1 来接收数据 1
            dat+=1;
        while(DATA);          //等待数据线拉低
    }
    return dat;
}
/**************************************************************************************
***********
函数名：接收 DHT11 的 40 位的数据并校验
调  用：? = DHT11_receive();
参  数：无
返回值：无
结  果：结果保存到 dat_r[i]中
备  注：
**************************************************************************************
***********/
void DHT11_receive()      //接收 40 位的数据
{
    uchar R_H,R_L,T_H,T_L,RH,RL,TH,TL,revise;
    DHT11_start();                // 主机控制 DHT11 开始工作
    if(DATA==0)
    {
        while(DATA==0);           //等待拉高
        Delay_Us(80);
        R_H=DHT11_rec_byte();    //接收湿度高八位
        R_L=DHT11_rec_byte();    //接收湿度低八位
        T_H=DHT11_rec_byte();    //接收温度高八位
        T_L=DHT11_rec_byte();    //接收温度低八位
        revise=DHT11_rec_byte(); //接收校正位[/i][/color][/font][/u][i]
        [u][font=宋体][color=#336699]       Delay30us();    //结束[/color][/font][/
u]
        [u][font=宋体][color=#336699]       if((R_H+R_L+T_H+T_L)==revise)   //校正
        {
            RH=R_H;
            RL=R_L;
            TH=T_H;
            TL=T_L;
        }
        /*数据处理，方便显示*/
        dat_r[0]='0'+(RH/10);
        dat_r[1]='0'+(RH%10);
        dat_r[2]='0'+(TH/10);
        dat_r[3]='0'+(TH%10);
    }
}
/*========串口初始化函数================================================*/
void UartInit()                                 //9600bps@11.0592MHz
{
    SCON = 0x50;                //8 位数据,可变波特率
    AUXR |= 0x04;
//AUXR |= 0x01;             //串口 1 选择定时器 2 为波特率发生器
    T2L =0xE0;   // 65536-(11059200/4/9600);                  //设定定时初值
    T2H =0xFE;   // (65536-(11059200/4/9600))>>8;             //设定定时初值
    AUXR |= 0x10;
}
/*===========串口发送函数(字节)=========================================*/
void Uartsend_1(unsigned char send)
{
    SBUF=send;                                                  //发送数据
    while(!TI);                                                 //等待前一帧数据发送完
毕
    TI=0;                                                       //复位标志位
}
//----------串口发送函数（字符串）----------------------------------------
void PrintString(unsigned char code *puts)    //发送一串字符串
{
for (; *puts != 0;
puts++)  Uartsend_1(*puts);         //遇到停止符 0 结束
}
/**************************************************************************************
***********
函数名：主函数
调  用：无
参  数：无
返回值：无
结  果：读 DHT11 数据并送到 1602 显示
备  注：
**************************************************************************************
***********/
/*void main (void)
{
    UartInit();
    Delay20ms();
    while(1)
    {
        Delay1s();            //经测试，两次连读要至少延时 80ms
        DHT11_receive();//接受数据
        //RH:XX%
        PrintString("RH:");
        Uartsend_1(dat_r[0]);
        Uartsend_1(dat_r[1]);
        PrintString("%;  ");
        //TMP:XXC
```

```
        PrintString("Tempreture:");
        Uartsend_1(dat_r[2]);
        Uartsend_1(dat_r[3]);
        Uartsend_1(0xdf);
        PrintString("C  ");
        Delay1s();


    }
}*/
```

6.nexie//数码管

```
#ifndef ___ NIXIE_H__
#define __NIXIE_H__
void Nixie(unsigned char Location,Number);
#endif
#include <regx52.h>
#include "Delay.h"
sbit WE=P2^0;//自行定义引脚
sbit DU=P2^1;
unsigned char code LEDchar[]={
        0xC0,0XF9,0XA4,0XB0,0X99,0X92,0X82,0XF8,
        0X80,0X90,0X88,0X83,0XC6,0XA1,0X86,0X8E
};
void Nixie(unsigned char Location,Number)
{
        switch(Location)
        {
                case 1:WE=1;P0=~(0X01<<0);WE=0;break;
                case 2:WE=1;P0=~(0X01<<1);WE=0;break;
                case 3:WE=1;P0=~(0X01<<2);WE=0;break;
                case 4:WE=1;P0=~(0X01<<3);WE=0;break;
                case 5:WE=1;P0=~(0X01<<4);WE=0;break;
                case 6:WE=1;P0=~(0X01<<5);WE=0;break;
                case 7:WE=1;P0=~(0X01<<6);WE=0;break;
                case 8:WE=1;P0=~(0X01<<7);WE=0;break;

        }
        DU=1;
                        P0=~LEDchar[Number];
                        DU=0;
            Delay(2);
}
```

7.ds18b20//温度传感器

ds18b20.h

```
/*********************************************************************************
**********************
** 工程功能 : 温度计DS18B20        头文件
** 工程作者 : Blue Sky Teams——WCW
** 工程版本 : V1.0
*********************************************************************************
**********************/
#ifndef _DS18B20_H_
#define _DS18B20_H_

  #ifndef uchar
        #define uchar unsigned char
  #endif

  #ifndef uint
        #define uint  unsigned int
  #endif

#include"intrins.h"
#include <INTRINS.H>
#include "STC12C5A60S2.H"

sbit DQ = P1^7;                         //DS18B20 的数据口位 P1^1

extern uint TPH;                        //存放温度值的高字节
extern uint TPL;                        //存放温度值的低字节
extern float TP;                                                        //存放温度值的
十进制数


/*********************************************************************************
**********************
** 函数功能 : 延时函数
** 函数说明 : 利用软件延时，占用CPU，经调试最小单位大约为1us
** 入口参数 : time:需要延时的时间，单位us
** 出口参数 : 无
*********************************************************************************
**********************/
extern void Delay_Us(uchar n);


/*********************************************************************************
**********************
** 函数功能 : DS18B20 复位函数
** 函数说明 : 复位DS18B20,并检测设备是否存在
** 入口参数 : 无
** 出口参数 : 无
*********************************************************************************
**********************/
extern void DS18B20_Reset();


/*********************************************************************************
**********************
** 函数功能 : DS18B20 读字节函数
** 函数说明 : 从DS18B20 读1 字节数据
** 入口参数 : 无
** 出口参数 : 从DS18B20 读回的1 字节数据
```

```
*************************************************************************************
**********************/
extern uchar DS18B20_ReadByte();


/*************************************************************************************
**********************
** 函数功能 : DS18B20 写字节函数
** 函数说明 : 向DS18B20 写1 字节数据
** 入口参数 : 要写入DS18B20 的1 字节数据
** 出口参数 : 无
*************************************************************************************
**********************/
extern void DS18B20_WriteByte(uchar dat);


/*************************************************************************************
**********************
** 函数功能 : DS18B20 开始转化温度数据
** 函数说明 : 到转化结束需要一定时间，否则读取的还是旧数据
** 入口参数 :
** 出口参数 :
*************************************************************************************
**********************/
extern void DS18B20_Start();

/*************************************************************************************
**********************
** 函数功能 : 读取并计算 DS18B20 的温度
** 函数说明 : 到转化结束需要一定时间，否则读取的还是旧数据
** 入口参数 :
** 出口参数 :
*************************************************************************************
**********************/
extern void DS18B20_End();

#endif
```

ds18b20.c

```
/*************************************************************************************
**********************
** 工程功能 : 温度计DS18B20       头文件
** 工程作者 : Blue Sky Teams——WCW
** 工程版本 : V1.0
*************************************************************************************
**********************/

#include"DS18B20.h"

uint TPH =0;                            //存放温度值的高字节
uint TPL =0;                            //存放温度值的低字节
float TP=0;                                                     //存放温度值的十进
```

制数

```
/*************************************************************************************
**********************
** 函数功能 : DS18B20 复位函数
** 函数说明 : 复位DS18B20,并检测设备是否存在
** 入口参数 : 无
** 出口参数 : 无
*************************************************************************************
**********************/
void DS18B20_Reset()
{
    uchar flag;

    DQ = 0;                     //送出低电平复位信号
    Delay_Us(250);              //延时至少480us
    Delay_Us(250);
    DQ = 1;                     //释放数据线
    Delay_Us(15);               //等待60us
        while(flag)
        flag = DQ;              //检测存在脉冲

    Delay_Us(250);              //等待设备释放数据线
    Delay_Us(250);
        Delay_Us(250);
}

/*************************************************************************************
**********************
** 函数功能 : DS18B20 读字节函数
** 函数说明 : 从DS18B20 读1 字节数据
** 入口参数 : 无
** 出口参数 : 从DS18B20 读回的1 字节数据
*************************************************************************************
**********************/
uchar DS18B20_ReadByte()
{
    uchar i;
    uchar dat = 0;

    for (i=0; i<8; i++)             //8 位计数器
    {
        DQ = 0;                     //开始时间片
        Delay_Us(1);                //延时等待
        DQ = 1;                     //准备接收
        Delay_Us(6);                //接收延时

        if (DQ) dat |= 0x80;        //读取数据
                dat >>=1;
        Delay_Us(50);               //等待时间片结束
    }
```

```
    return dat;
}

/*******************************************************************************
**********************
** 函数功能 : DS18B20 写字节函数
** 函数说明 : 向DS18B20 写1 字节数据
** 入口参数 : 要写入DS18B20 的1 字节数据
** 出口参数 : 无
*******************************************************************************
**********************/
void DS18B20_WriteByte(uchar dat)
{
    char i;

    for (i=0; i<8; i++)             //8 位计数器
    {
        DQ = 0;                     //开始时间片
        Delay_Us(1);                //延时等待
        DQ= dat & 0x01;                             //送出数据
        Delay_Us(60);               //等待时间片结束
        DQ = 1;                     //恢复数据线
        Delay_Us(1);                //恢复延时
                dat >>= 1;
    }
}

/*******************************************************************************
**********************
** 函数功能 : DS18B20 开始转化温度数据
** 函数说明 : 到转化结束需要一定时间，否则读取的还是旧数据
** 入口参数 :
** 出口参数 :
*******************************************************************************
**********************/
void DS18B20_Start()
{
        DS18B20_Reset();                //设备复位
        DS18B20_WriteByte(0xCC);        //跳过ROM 命令
        DS18B20_WriteByte(0x44);         //开始转换命令
}


/*******************************************************************************
**********************
** 函数功能 : 读取并计算 DS18B20 的温度
** 函数说明 : 到转化结束需要一定时间，否则读取的还是旧数据
** 入口参数 :
** 出口参数 :
*******************************************************************************
**********************/
```

```
void DS18B20_End()
{
        DS18B20_Reset();                //设备复位
        DS18B20_WriteByte(0xCC);        //跳过ROM 命令
        DS18B20_WriteByte(0xBE);        //读暂存存储器命令
        TPL = DS18B20_ReadByte();       //读温度低字节
        TPH = DS18B20_ReadByte();       //读温度高字节

        TP=((TPH<<8)|TPL)*0.0625;                       //将读取的数据转换成十进制数
}
```

8.ds1302//时钟

ds1302.h

```
#ifndef _DS1302_H_
#define _DS1302_H_

  #ifndef uchar
        #define uchar unsigned char
  #endif

  #ifndef uint
        #define uint  unsigned int
  #endif

#include"intrins.h"
#include "STC12C5A60S2.H"


sbit DS1302_SCLK =P1^5;  //时钟
sbit DS1302_IO   =P1^6;  //数据输入输出
sbit DS1302_RST  =P1^7;  //复位/片选线

typedef struct
        {
                uchar SEC;                      //00~59
                uchar MIN;                      //00~59
                uchar HR;                       //00-23
                uchar DATE;                     //00-31
                uchar MONTH;            //01-12
                uchar DAY;                      //01-07
                uchar YEAR;                     //00-99
        }TIME_STRUCT;
extern TIME_STRUCT TIME;//初始化时间参数

extern uchar dtime[7][2]; //保存的时间 字符


/*********************************************************************
//函数名称: DS1302_WByte
```

```
//函数功能: 往DS1302 写入 1 Byte 数据
//输入值:   ndata: 寄存器的数据或地址
//返回值:   无
*************************************************************************/
extern void DS1302_WByte(uchar ndata);

/************************************************************************
//函数名称: DS1302_RByte
//函数功能: 从DS1302 读取 1 Byte 数据
//输入值:   无
//返回值:   ndata:读取的数据
*************************************************************************/
extern  uchar DS1302_RByte(void);

/************************************************************************
//函数名称: DS1302_Wdata
//函数功能: 往DS1302 某地址写入某数据
//输入值:   nAddr: DS1302 地址, ndata: 要写的数据
//返回值:   无
*************************************************************************/
extern void DS1302_Wdata(uchar nAddr, uchar ndata);

/************************************************************************
//函数名称: DS1302_Rdata
//函数功能: 从DS1302 某地址读取数据
//输入值:   nAddr: DS1302 地址
//返回值:   ndata: 读取的数据
*************************************************************************/
extern uchar DS1302_Rdata(uchar nAddr);

/************************************************************************
//函数名称: InitDS1302
//函数功能: DS1302 初始时间设定
//输入值:   无
//返回值:   无
*************************************************************************/
extern void InitDS1302(void);

/************************************************************************
//函数名称: GetDS1302
//函数功能: DS1302 当前时间读取
//输入值:   无
//返回值:   无
*************************************************************************/
 extern  void GetDS1302(void);

#endif

 /*
 main()
{
   InitLCD();       //初始化 1602
   InitDS1302();   //测试断开电源时，电池供电功能
   while(1)            //进入死循环，防止看门狗复位
   {
    GetDS1302();   //提取时间参数
    xian1();        //显示第一行
    xian2();        //显示第二行
   }
}
*/

ds1302.c
#include"ds1302.h"
TIME_STRUCT  TIME =
        {
                0x00,           //秒   00~59
                0x10,           //分   00~59
                0x00,           //时   00-23
                0x10,           //日   00-31
                0x05,           //月   01-12
                0x05,           //星期 01-07
                0x19,           //年   00-99
        };
        //初始化时间参数

uchar dtime[7][2]={
                    {0x20,0x20},        //秒 十位和个位
                                         {0x20,0x20},    //分 十位和个位
                                         {0x20,0x20},    //时 十位和个位
                                         {0x20,0x20},    //日 十位和个位
                                         {0x20,0x20},    //月 十位和个位
                                         {0x20,0x20},    //周 十位和个位
                                         {0x20,0x20}     //年 十位和个位
                          };
/************************************************************************
//函数名称: DS1302_WByte
//函数功能: 往 DS1302 写入 1 Byte 数据
//输入值:   ndata: 寄存器的数据或地址
//返回值:   无
*************************************************************************/
  void DS1302_WByte(uchar ndata)
{
  uchar i;
  for(i=8; i>0; i--)                    //循环 8 次写入 8 位数据
  {
    DS1302_IO = (bit)(ndata&0x01);      //取最低位数据，从 0 位至 7 位依次传送
    DS1302_SCLK = 1;                    //给一个脉冲，将数据写入 1302
    _nop_();
    DS1302_SCLK = 0;
    ndata>>=1;                          //即 ndata = ndata >> 1;
  }
```

```
}
/*********************************************************************
//函数名称: DS1302_RByte
//函数功能: 从 DS1302 读取 1 Byte 数据
//输入值:   无
//返回值:   ndata:读取的数据
*********************************************************************/
  uchar DS1302_RByte(void)
{
   uchar i;
   uchar ndata=0;
   for(i=8;i>0;i--)                 //循环 8 次读出 8 位数据
        {
         DS1302_IO=1;                    //初始化数据 IO
         ndata>>=1;                              //即 ndata = ndata >> 1;
         if(DS1302_IO) ndata|=0x80;      //从数据口读取 1 位数据
     DS1302_SCLK = 1;                //给一个脉冲
     _nop_();
     DS1302_SCLK = 0;

         }
     return (ndata);                 //返回结果
}
/*********************************************************************
//函数名称: DS1302_Wdata
//函数功能: 往 DS1302 某地址写入某数据
//输入值:   nAddr: DS1302 地址, ndata: 要写的数据
//返回值:   无
*********************************************************************/
void DS1302_Wdata(uchar nAddr, uchar ndata)
{
  DS1302_RST=0;
  DS1302_SCLK=0;
  DS1302_RST=1;
  DS1302_WByte(nAddr);       // 写 1Byte 地址
  DS1302_WByte(ndata);       // 写 1Byte 数据
  DS1302_SCLK=1;
  DS1302_RST=0;
}
/*********************************************************************
//函数名称: DS1302_Rdata
//函数功能: 从 DS1302 某地址读取数据
//输入值:   nAddr: DS1302 地址
//返回值:   ndata: 读取的数据
*********************************************************************/
uchar DS1302_Rdata(uchar nAddr)
{
    uchar ndata;
        DS1302_RST=0;
        DS1302_SCLK=0;
        DS1302_RST=1;
    DS1302_WByte(nAddr);       /* 地址，命令 */
    ndata = DS1302_RByte();    /* 读 1Byte 数据 */
    DS1302_SCLK=1;
        DS1302_RST=0;
    return(ndata);
}
/*********************************************************************
//函数功能: DS1302 初始时间设定
*********************************************************************/
void InitDS1302(void)
{
        DS1302_Wdata(0x8e,0x00); //控制命令,WP=0,写操作
        DS1302_Wdata(0x90,0xa5);
        /*
        地址 0x90 为充电寄存器，可以对充电电流进行限制，写入
        内容高 4 位固定为 1010（其他组合均不能充电），低 4
        位的首 2 位是选择内部降压二极管的个数的，01 代表在
        充电回路串入 1 个二极管，10 代表串入 2 个；最后 2 位可
        设定串入的电阻的数值：01 为 2k 欧，10 为 4k 欧，11 为 8k 欧。
        */
        DS1302_Wdata(0x80,TIME.SEC);  //秒
        DS1302_Wdata(0x82,TIME.MIN);  //分
        DS1302_Wdata(0x84,TIME.HR);   //时
        DS1302_Wdata(0x86,TIME.DATE); //日
        DS1302_Wdata(0x88,TIME.MONTH);//月
        DS1302_Wdata(0x8a,TIME.DAY);  //星期
        DS1302_Wdata(0x8c,TIME.YEAR); //年

        DS1302_Wdata(0x8e,0x80); //控制命令,WP=1,写保护
}
/*********************************************************************
//S1302 当前时间读取
*********************************************************************/
void GetDS1302(void)
{
        TIME.SEC= DS1302_Rdata(0x81);    //从 DS1302 读取秒数据
    dtime[0][0]=(TIME.SEC>>4)+0x30;      //十位
        dtime[0][1]=(TIME.SEC&0x0F)+0x30;    //个位

        TIME.MIN= DS1302_Rdata(0x83);    //从 DS1302 读取分数据
    dtime[1][0]=(TIME.MIN>>4)+0x30;      //十位
        dtime[1][1]=(TIME.MIN&0x0F)+0x30;    //个位

        TIME.HR         = DS1302_Rdata(0x85);    //从 DS1302 读取时数据
    dtime[2][0]=(TIME.HR>>4)+0x30;       //十位
        dtime[2][1]=(TIME.HR&0x0F)+0x30;     //个位

        TIME.DATE       = DS1302_Rdata(0x87);    //从 DS1302 读取日数据
    dtime[3][0]=(TIME.DATE>>4)+0x30;     //十位
        dtime[3][1]=(TIME.DATE&0x0F)+0x30;   //个位

        TIME.MONTH      = DS1302_Rdata(0x89);    //从 DS1302 读取月数据
```

```
    dtime[4][0]=(TIME.MONTH>>4)+0x30;    //十位
        dtime[4][1]=(TIME.MONTH&0x0F)+0x30;  //个位

        TIME.DAY= DS1302_Rdata(0x8b);    //从 DS1302 读取星期数据
    dtime[5][0]=(TIME.DAY>>4)+0x30;      //十位
        dtime[5][1]=(TIME.DAY&0x0F)+0x30;    //个位

        TIME.YEAR       = DS1302_Rdata(0x8d);    //从 DS1302 读取年数据
    dtime[6][0]=(TIME.YEAR>>4)+0x30;     //十位
        dtime[6][1]=(TIME.YEAR&0x0F)+0x30;   //个位
}
 /*
                GetDS1302(); //更新时钟数据
                chs[0]= dtime[2][0]; //时十位
                chs[1]= dtime[2][1]; //时个位
                chs[2]=':';
                chs[3]= dtime[1][0]; //分十位
                chs[4]= dtime[1][1]; //分十位
                chs[5]=':';
                chs[6]= dtime[0][0]; //分十位
                chs[7]= dtime[0][1]; //分十位
                chs[8]=0;                           //结尾
                Hanzi_Disp(1,0,chs);
 */
```

9.hx711 称重

hx711.h

```
#ifndef __HX711_H__
#define __HX711_H__


#include "STC12C5A60S2.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
```

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
}
```

10.外部中断

irq_ext.h

```
/*********************************************************************************
*****************
工程名称：        IRQ_ext
功能描述：        按下按键 S17，LED 发光二极管加 1
硬件连接：   用 8 位杜邦线将 J8 与 J13 连接,用 1 位杜邦线将 J7_S17 与 J9_2 连接
维护记录：   2011-8-22
*********************************************************************************
***************/
#ifndef   _irq_ext_H_
#define   _irq_ext_H_

#include "STC12C5A60S2.H"      //包含头件

  #ifndef uchar
        #define uchar unsigned char
  #endif

  #ifndef uint
        #define uint  unsigned int
  #endif

#endif
```

irq_ext.c

```
/*********************************************************************************
*****************
工程名称：        IRQ_ext
功能描述：        按下按键 S17，LED 发光二极管加 1
硬件连接：   用 8 位杜邦线将 J8 与 J13 连接,用 1 位杜邦线将 J7_S17 与 J9_2 连接
维护记录：   2011-8-22
*********************************************************************************
***************/

#include "int_it.h"

//main()
//{
//  IT0=1;       //下降沿触发
//  EX0=1;       //开中断 0
//  EA=1;        //开总中断

  //IT1=1;       //下降沿触发
  //EX1=1;       //开中断 1
  //EA=1;        //开总中断
//
//  while(1);    //等待中断产生，按下 S17 按键产生中断信号
//}


//*********************************************************************************
*****************
//外部中断 0 中断子程序
//*********************************************************************************
*****************
void Int0(void) interrupt 0
{

}

//*********************************************************************************
*****************
//外部中断 1 中断子程序
//*********************************************************************************
*****************
void Int1(void) interrupt 2
{

}
```

10.io_expand//IO 扩展

io_expand.h

```
#ifndef _IO_EXPAND_H_
#define _IO_EXPAND_H_

#include "STC12C5A60S2.H"      //包含头文件

  #ifndef uchar
        #define uchar unsigned char
  #endif

  #ifndef uint
        #define uint  unsigned int
  #endif


//74HC595 引脚定义
sbit SH_CP=P1^5;  //时钟
sbit ST_CP=P1^6;  //上升沿更新数据
sbit DS=P1^7;     //数据

//74HC138 引脚定义
sbit HC138_A=P1^0;
sbit HC138_B=P1^1;
sbit HC138_C=P1^2;
sbit HC138_EN=P1^3;
```

```
extern void delay(uint time);

//*****************************************************************************
*****************
//向74HC595 写一字节数据
//*****************************************************************************
*****************
extern void wbyte_595(uchar temp);

//*****************************************************************************
*****************
//向74HC138 写位选数据 0—7   大于7 关闭
//*****************************************************************************
*****************
extern void wbyte_138(uchar temp);

#endif
```

io_expand.c

```
#include "IO_expand.h"
//*****************************************************************************
*****************
//向74HC595 写一字节数据
//*****************************************************************************
*****************
void wbyte_595(uchar temp)
 {
     uchar i;                       //定义循环变量
     ST_CP=0;                       //置为低电平
     for(i=0;i<8;i++)               //循环8 次，写入1 字节
      {
       SH_CP=0;                     //时钟置为低电平
       if((temp & 0x80)!=0)  DS=1; //如果最高位为1，发送数据1
         else DS=0;                 //如果最高位为0，发送数据0
       SH_CP=1;                     //数据在SHcp 的上升沿输入到移位寄存器中
       temp<<=1;                    //左移1 位，准备写入下1 位数据
      }
    ST_CP=1;                        //上升沿时移位寄存器的数据进入数据存储寄存器
    delay(10);                      //延时
 }


//*****************************************************************************
*****************
//138 低电平位置 0-7   大于7 关闭
//*****************************************************************************
*****************
void wbyte_138(uchar temp)
{
        if(temp>7){
                HC138_EN=0;
                return ;
        }

        HC138_EN=1;
        HC138_A = temp;           //最低位
        HC138_B = temp>>1;
        HC138_C = temp>>2;        //最高位

}
```

11.独立键盘和矩阵按键

key.h

```
/*****************************************************************************
****************
工程名称:        key

维护记录:   2011-8-22
*****************************************************************************
***************/
#ifndef _KEY_H_
#define _KEY_H_

#include "STC12C5A60S2.H"      //包含头文件
#ifndef uchar
        #define uchar unsigned char
#endif
#ifndef uint
        #define uint  unsigned int
#endif
#define GPIO_KEY P3

sbit key1=P3^0;
sbit key2=P3^1;
sbit key3=P3^2;
sbit key4=P3^3;

extern uchar KeyValue ;
extern void Delay_Ms(uint time);
//*****************************************************************************
****************
//矩阵键盘扫描函数   需要delayms()函数
//*****************************************************************************
****************
extern void KeyDown(void);
extern unsigned char KeyScan();
#endif
```

key.c

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

```
库名称：  key
作者：  Jun
日期：   2023-4-21
**************************************************************************************
***************/
#include "key.h"      //包含头文件
uchar KeyValue = 16;



//**************************************************************************************
*****************
//矩阵键盘扫描函数
//**************************************************************************************
*****************
void KeyDown(void){
        char a=0;
        GPIO_KEY=0x0f;
        if(GPIO_KEY!=0x0f){ //读取按键是否按下
                Delay_Ms(10);//延时10ms 进行消抖
                if(GPIO_KEY!=0x0f){//再次检测键盘是否按下
                        //测试列
                        GPIO_KEY=0X0F;
                        switch(GPIO_KEY){
                                case(0X07):     KeyValue=0;break;
                                case(0X0b):     KeyValue=1;break;
                                case(0X0d): KeyValue=2;break;
                                case(0X0e):     KeyValue=3;break;
                        }
                        //测试行
                        GPIO_KEY=0XF0;
                        switch(GPIO_KEY){
                                case(0X70):     KeyValue=KeyValue;break;
                                case(0Xb0):     KeyValue=KeyValue+4;break;
                                case(0Xd0): KeyValue=KeyValue+8;break;
                                case(0Xe0):     KeyValue=KeyValue+12;break;
                        }
                        while((a<50)&&(GPIO_KEY!=0xf0)){  //检测按键松手检测
                                Delay_Ms(10);
                                a++;
                        }
                }
        }
}
/**
  * @brief  获取独立按键键码
  * @param  无
  * @retval 按下按键的键码，范围：0~4，无按键按下时返回值为0
  */
unsigned char KeyScan(){
        unsigned char KeyNumber=0;
        if(key1==0){Delay_Ms(20);while(key1==0);Delay_Ms(20);KeyNumber=1;}
        if(key2==0){Delay_Ms(20);while(key2==0);Delay_Ms(20);KeyNumber=2;}
        if(key3==0){Delay_Ms(20);while(key3==0);Delay_Ms(20);KeyNumber=3;}
        if(key4==0){Delay_Ms(20);while(key4==0);Delay_Ms(20);KeyNumber=4;}
        return KeyNumber;
}
```

12.lcd1602 显示屏

lcd1602.h

```
/**************************************************************************************
**********************
** 文件功能 : LCD1602 驱动程序
** 工程作者 : Blue Sky Teams——ZZL
** 工程版本 : V1.0
**************************************************************************************
**********************/

#ifndef _LCD1602_H_
#define _LCD1602_H_

#include <intrins.h>
#include "STC12C5A60S2.H"

  #ifndef uchar
        #define uchar unsigned char
  #endif

  #ifndef uint
        #define uint  unsigned int
  #endif

/**************************************************************************************
**********************
** LCD1602 接口定义
**************************************************************************************
**********************/
sbit RS_1602 = P2^5;     //数据命令选择端
sbit RW_1602 = P2^4;     //写选择端
sbit EN_1602 = P2^3;     //使能信号
#define LCD_PORT P0      //LCD1602 数据接口

/**************************************************************************************
**********************
** LCD1602 宏定义指令集
** 指令集的指令作为 void Write_1602_Com(unsigned char zhiling) 参数发送给 LCD1602 之后可
以让液晶 LCD1602 执行相应的功能，
** 使用者可根据自己要实现的功能选择相应的指令，具体说明请参看文档“LCD1602 液晶完整中文资料”
有关指令的章节
**************************************************************************************
**********************/
#define CLEAR_SCREEN                    0X01            //清屏
#define CURSOR_RESET                    0X02            //光标复位
```

```
/*****************************************输入方式设置*****************************
**********************/
#define SET_MOD                                 0X04            //配合一下两位来配
置模式

#define SET_MOD_AC_ADD                  0X02            //数据读写操作后AC 自加一
#define SET_MOD_AC_DEC                  0X00            //数据读写操作后AC 自减一
#define SET_MOD_MOVE_ON                 0X01            //数据读写操作后画面移动
#define SET_MOD_MOVE_OFF        0X00            //数据读写操作后画面不动

/*****************************************显示开关控制*****************************
**********************/
#define DISPLAY_SET                             0X08            //配合下面三位来配
置模式

#define DISPLAY_SET_ON                  0X04            //显示开
#define DISPLAY_SET_OFF                 0X00            //显示关
#define DISPLAY_SET_CURSOR_ON   0X02            //光标显示开
#define DISPLAY_SET_CURSOR_OFF  0X00            //光标显示关
#define DISPLAY_SET_BLINK_ON    0X01            //光标闪烁
#define DISPLEY_SET_BLINK_OFF   0X00            //光标不闪烁

/*****************************************光标，画面位移*****************************
**********************/
#define COURSOR_SHIFT_LEFT              0X10            //光标左移一个字符位，AC 减
一
#define COURSOR_SHIFT_RIGHT             0X14            //光标右移一个字符位，AC 加
一
#define FRAME_SHIFT_LEFT        0X18            //画面左移一个字符位，光标不变
#define FRAME_SHIFT_RIGHT               0X1C            //画面右移一个字符位，光标不
变

/*****************************************显示功能设置*****************************
**********************/
#define DISPLAY_MOD                             0X30            //默认设置为8 位数
据口，配合一下两个位来时配置模式
#define DISPLAY_MOD_TWO_LINE    0X08            //两行显示
#define DISPLAY_MOD_ONE_LINE    0X00            //一行显示
#define DISPLAY_MOD_5_10        0X04            //5*10 的点阵字符显示
#define DISPLAY_MOD_5_7                 0X00            //5*7 的点阵字符显示


/*****************************************CGRAM 起始地址*****************************
**********************/

//uchar code CCGRAM_ADD = 0X40;

/*******************************************************************************
**********************
** 函数功能 ： 读写 LCD1602 时用于读写时序的时钟延时
** 入口参数 ： z：延时长度
** 出口参数 ： 无
*******************************************************************************
**********************/
extern void LCD_Delay(unsigned char z);


/*******************************************************************************
**********************
** 函数功能 ： LCD1602 忙碌查询
** 函数说明 ： 每次读写液晶时需要事先检测芯片是否处于忙碌状态
** 入口参数 ： 无
** 出口参数 ： 返回值为1：等待超时
**             返回值为0：芯片处于空闲状态
*******************************************************************************
**********************/
extern unsigned char Check_1602_Busy(void);


/*******************************************************************************
**********************
** 函数功能 ： 向液晶写入指令
** 函数说明 ：
** 入口参数 ： 待写入的指令
** 出口参数 ： 无
*******************************************************************************
**********************/
extern void Write_1602_Com(unsigned char zhiling);//写指令


/*******************************************************************************
**********************
** 函数功能 ： 向液晶写入数据
** 函数说明 ：
** 入口参数 ： 待写入的数据，数据为 LCD1062 用来做显示用的，比如说 shuju = 0x30，则显示“0”，
**                                 现成的字符数据可查询"LCD1602 液晶完整中文资料.pdf"中“CGROM 中
字符码与字字符字模关系对照表”
** 出口参数 ： 无
*******************************************************************************
**********************/
extern void Write_1602_Data(unsigned char shuju);//写数据


/*******************************************************************************
**********************
** 函数功能 ： 在指定位置连续写入一串字符
** 函数说明 ： 注意写入字符串的长度，不要超过屏幕单行的显示范围
** 入口参数 ： str：    待写入字符串首地址
**             addr：   写入液晶显示的地址
** 出口参数 ： 无
*******************************************************************************
```

```
**********************/
extern void Write_1602_String(unsigned char *str,unsigned char addr);


/*****************************************************************************
***********************
** 函数功能 : 在指定位置写入一个整数
** 函数说明 : 整数的范围从0~65535，更大的需要采用long 整型的进行扩展，注意整数的位数和地
址不要超出显示范围
** 入口参数 : number：待写入的整数
**            addr：   写入液晶显示的地址
** 出口参数 : 无
*****************************************************************************
**********************/
extern void Write_Num(unsigned int number,unsigned char addr);



/*****************************************************************************
***********************
** 函数功能 : 液晶初始化
** 函数说明 : 用液晶之前需要先初始化液晶，配置对应的显示模式
** 入口参数 : time:需要延时的时间，单位ms
** 出口参数 : 无
*****************************************************************************
**********************/
extern void LCD_1602_Init(); //初始化



#endif

/*
void main()
{
        uchar i,temp;
        LCD_1602_Init(); //液晶显示前进行初始化
        Write_1602_String("I LIKE MCU 2012",0X80);
        Write_1602_Data(0x05);//自定义字符'年'
        Write_1602_String("By Blue Sky",0XC5);
        while(1)
        {
                if(temp == 0)
                {
                        i++;
                        Delay_Ms(500);
                        Write_1602_Com(FRAME_SHIFT_LEFT);//画面左移一个字符位

                        Write_1602_String("I LIKE MCU 2012",0x80+i);
                        Write_1602_Data(0x05);
                        if(i == 5)temp = 1;
                }
                else
                {
                        i--;
                        Delay_Ms(500);
                        Write_1602_Com(FRAME_SHIFT_RIGHT);//画面右移一个字符位

                        Write_1602_String("I LIKE MCU 2012",0x80+i);
                        Write_1602_Data(0x05);
                        if(i == 0)temp = 0;
                }
        }
}

*/
```

lcd1602.c

```
/*****************************************************************************
***********************
** 文件功能 : LCD1602 驱动程序
** 工程作者 : Blue Sky Teams——ZZL
** 工程版本 : V1.0
*****************************************************************************
**********************/
#include "LCD1602.h"

/*****************************************************************************
***********************
** 函数功能 : 读写LCD1602 时用于读写时序的时钟延时
** 入口参数 : z：延时长度
** 出口参数 : 无
*****************************************************************************
**********************/
void LCD_Delay(unsigned char z)
{
        unsigned char x,y;
        for(x = z;x > 0;x --)
                for(y = 50;y>0;y--);
}

/*****************************************************************************
***********************
** 函数功能 : LCD1602 忙碌查询
** 函数说明 : 每次读写液晶时需要事先检测芯片是否处于忙碌状态
** 入口参数 : 无
** 出口参数 : 返回值为1：等待超时
**            返回值为0：芯片处于空闲状态
*****************************************************************************
**********************/
unsigned char Check_1602_Busy(void)
{
        unsigned int time=0;
        RS_1602 = 0;
```

```
        RW_1602 = 1;
        EN_1602 = 1;
        while( (LCD_PORT&0X80) != 0X00)
        {
                time ++;
                if(time > 1000) return 1;//实际测量会遇到有些型号的液晶BF 一直为高，如果等待超时跳出循环
        }
        return 0;
}

/*********************************************************************************************************
** 函数功能 : 向液晶写入指令
** 函数说明 :
** 入口参数 : 待写入的指令
** 出口参数 : 无
*********************************************************************************************************/
void Write_1602_Com(unsigned char zhiling)//写指令
{
        Check_1602_Busy();      //读忙碌状态，等待液晶处于空闲
        RS_1602 = 0;            //RS_1602 为低电平
        RW_1602 = 0;            //RW_1602 为低电平
        EN_1602 = 1;
        LCD_PORT = zhiling;     //准备好数据口的数据
        LCD_Delay(1);           //根据时序要求，延时一段时间，等待数据口数据稳定
        EN_1602 = 0;            //产生一个下降沿，将数据口的数据读入到液晶中
        LCD_Delay(1);

}

/*********************************************************************************************************
** 函数功能 : 向液晶写入数据
** 函数说明 :
** 入口参数 : 待写入的数据，数据为 LCD1062 用来做显示用的，比如说 shuju = 0x30，则显示“0”，
**                          现成的字符数据可查询"LCD1602 液晶完整中文资料.pdf"中“CGROM 中字符码与字字符字模关系对照表”
** 出口参数 : 无
*********************************************************************************************************/
void Write_1602_Data(unsigned char shuju)//写数据
{
        Check_1602_Busy();
        RS_1602 = 1;
        RW_1602 = 0;
        EN_1602 = 1;
        LCD_PORT = shuju;
        LCD_Delay(1);
        EN_1602 = 0;
        LCD_Delay(1);
}

/*********************************************************************************************************
** 函数功能 : 在指定位置连续写入一串字符
** 函数说明 : 注意写入字符串的长度，不要超过屏幕单行的显示范围
** 入口参数 : str:    待写入字符串首地址
**            addr:   写入液晶显示的地址
** 出口参数 : 无
*********************************************************************************************************/
void Write_1602_String(unsigned char *str,unsigned char addr)
{
        Write_1602_Com(addr);
        while(*str)
        {
                Write_1602_Data(*str);
                str++;
        }
}

/*********************************************************************************************************
** 函数功能 : 在指定位置写入一个整数
** 函数说明 : 整数的范围从0~65535，更大的需要采用Long 整型的进行扩展，注意整数的位数和地址不要超出显示范围
** 入口参数 : number：待写入的整数
**            addr:   写入液晶显示的地址
** 出口参数 : 无
*********************************************************************************************************/
void Write_Num(unsigned int number,unsigned char addr)
{
        Write_1602_Com(addr);
        if(number/10000 != 0)
        {
                Write_1602_Data(0x30 + number/10000);
                Write_1602_Data(0x30 + number%10000/1000);
                Write_1602_Data(0x30 + number%1000/100);
                Write_1602_Data(0x30 + number%100/10);
                Write_1602_Data(0x30 + number%10);
        }
        else if(number/1000 != 0)
        {
                Write_1602_Data(0x30 + number/1000);
                Write_1602_Data(0x30 + number%1000/100);
                Write_1602_Data(0x30 + number%100/10);
                Write_1602_Data(0x30 + number%10);
        }
        else if(number/100 != 0)
        {
                Write_1602_Data(0x30 + number/100);
```

```
                Write_1602_Data(0x30 + number%100/10);
                Write_1602_Data(0x30 + number%10);
        }
        else if(number/10 != 0)
        {
                Write_1602_Data(0x30 + number/10);
                Write_1602_Data(0x30 + number%10);
        }
        else Write_1602_Data(0x30 + number);
}


/*********************************************************************************
**********************
** 函数功能 : 液晶初始化
** 函数说明 : 用液晶之前需要先初始化液晶，配置对应的显示模式
** 入口参数 : time:需要延时的时间，单位ms
** 出口参数 : 无
*********************************************************************************
**********************/
void LCD_1602_Init()//初始化
{
        Write_1602_Com(DISPLAY_MOD | DISPLAY_MOD_TWO_LINE | DISPLAY_MOD_5_7);//设置
显示模式，两行显示，字符点阵大小为5*7:0X38

        Write_1602_Com(DISPLAY_SET | DISPLAY_SET_ON | DISPLAY_SET_CURSOR_OFF | DISP
LEY_SET_BLINK_OFF);//开显示，不显示光标，光标不闪烁:0X0F

        Write_1602_Com(SET_MOD | SET_MOD_AC_ADD | SET_MOD_MOVE_OFF);//设置写入数据后
指针AC 加一,画面不动：0X06

        Write_1602_Com(CLEAR_SCREEN);//清屏
        Write_1602_Com(CURSOR_RESET);
        Write_1602_Com(0x80);
}
```

13.mpu6050 姿态传感器

mpu6050.h

```
//
// Created by Jay on 2022/6/11.
//

#ifndef __MPU6050_H
#define __MPU6050_H

#include <STC12C5A60S2.H>
#include <math.h>    //Keil library
#include <stdio.h>   //Keil library
#include"intrins.h"
#include "delay.h"

#ifndef uchar
#define uchar unsigned char
#endif

#ifndef uint
#define uint  unsigned int
#endif

//****************************************
// 定义51单片机端口
//****************************************
#define DataPort P0             //LCD1602 数据端口
sbit    SCL=P1^0;                       //IIC 时钟引脚定义
sbit    SDA=P1^1;                       //IIC 数据引脚定义
//****************************************
// 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
//****************************************
//定义类型及变量
//****************************************

//int   Temperature,Temp_h,Temp_l;       //温度及高低位数据
//****************************************
//函数声明
//****************************************
void  delay(unsigned int k);
                        //延时
void  lcd_printf(uchar *s,int temp_data);
```

```
//MPU6050 操作函数
void  InitMPU6050();
                                        //初始化 MPU6050
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();

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 //__MPU6050_H
```

mpu6050.c

```
//
// Created by Jay on 2022/6/11.
//

#include "mpu6050.h"
void  SeriPushSend(uchar send_data)
{
    SBUF=send_data;
    while(!TI);TI=0;
}
//*************************************
//延时
//*************************************
/*void delay(unsigned int k)
{
    unsigned int i,j;
    for(i=0;i<k;i++)
    {
        for(j=0;j<121;j++);
    }
}*/

//*************************************
//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
//*************************************

//*************************************
//I2C 起始信号
//*************************************
void I2C_Start()
{
    SDA = 1;                    //拉高数据线
    SCL = 1;                    //拉高时钟线
    Delay_Us(5);                 //延时
    SDA = 0;                    //产生下降沿
    Delay_Us(5);                //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
    Delay_Us(5);                 //延时
    SDA = 1;                    //产生上升沿
    Delay_Us(5);                 //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
    Delay_Us(5);                 //延时
    SCL = 0;                    //拉低时钟线
    Delay_Us(5);                  //延时
}
//*************************************
//I2C 接收应答信号
//*************************************
bit I2C_RecvACK()
{
    SCL = 1;                    //拉高时钟线
    Delay_Us(5);                  //延时
    CY = SDA;                   //读应答信号
    SCL = 0;                    //拉低时钟线
    Delay_Us(5);                  //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
        Delay_Us(5);              //延时
        SCL = 0;                //拉低时钟线
        Delay_Us(5);             //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
        Delay_Us(5);              //延时
        dat |= SDA;             //读数据
        SCL = 0;                //拉低时钟线
        Delay_Us(5);              //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)
{
    uchar H,L;
    H=Single_ReadI2C(REG_Address);
    L=Single_ReadI2C(REG_Address+1);
    return (H<<8)+L;   //合成数据
}


/*void init_uart()
{
    TMOD=0x21;
    TH1=0xfd;
    TL1=0xfd;

    SCON=0x50;
    PS=1;      //串口中断设为高优先级别
    TR0=1;          //启动定时器
    TR1=1;
    ET0=1;     //打开定时器 0 中断
    ES=1;
    EA=1;
}*/

//*********************************************************
//主程序
//*********************************************************
/*
void main()
{
    delay(500);          //上电延时
//      InitLcd();              //液晶初始化
    init_uart();
    InitMPU6050();       //初始化 MPU6050
    delay(150);
    while(1)
    {
```

```
        Display10BitData(GetData(ACCEL_XOUT_H),2,0);      //显示X 轴加速度
        Display10BitData(GetData(ACCEL_YOUT_H),7,0);      //显示Y 轴加速度
        Display10BitData(GetData(ACCEL_ZOUT_H),12,0);     //显示Z 轴加速度
        Display10BitData(GetData(GYRO_XOUT_H),2,1);       //显示X 轴角速度
        Display10BitData(GetData(GYRO_YOUT_H),7,1);       //显示Y 轴角速度
        Display10BitData(GetData(GYRO_ZOUT_H),12,1);      //显示Z 轴角速度

        SeriPushSend(0x0d);
        SeriPushSend(0x0a);//换行，回车
        delay(100);
    }
}*/
```

14.lcd12864 显示屏

lcd12864.h

```
/*****************************************************************************
***********************
** 文件功能 : 12864 液晶并行驱动程序
** 注意事项 : 硬件部分需要将12864 的第15 个管脚通过电阻接到高电平
** 工程作者 : Blue Sky Teams——ZZL
** 工程版本 : V1.0
*****************************************************************************
*********************/
#ifndef _LCD12864_H_
#define _LCD12864_H_

  #ifndef uchar
        #define uchar unsigned char
  #endif

  #ifndef uint
        #define uint  unsigned int
  #endif

#include"intrins.h"
#include "STC12C5A60S2.H"
/*****************************************************************************
***********************
**            CE----------P0^5     OUT
**                        RW----------P0^6         OUT
**            RS----------P0^7     OUT
**            date--------P2 1~7  IN/OUT
*************************************液晶管脚接口定义*****************************
*************************/
sbit     CE     =P2^3;  //锁存控制
sbit     RW     =P2^4;  //读写口
sbit     RS             =P2^5;  //命令/数据选择

sbit RES = P2^0; //复位
sbit PSB = P2^2; //串并选择

#define  DATA_PORT      P0


#define basic_commod  0x30
#define extern_commod 0x34
#define cursor_on          0x0f
#define cursor_off         0x0c
#define clear              0x01
#define draw_on            0x36
#define draw_off  0x34


extern uchar code addr_tab[32];

 /*****************************************************************************
***********************
** 函数功能 : 延时函数
** 函数说明 : 利用软件延时，占用CPU，经调试最小单位大约为1us
** 入口参数 : time:需要延时的时间，单位us
** 出口参数 : 无
*****************************************************************************
*********************/
extern void Delay_Us(uchar n);


/*****************************************************************************
***********************
** 函数名称 : void Check_12864_Busy(void)
** 函数功能 : 读忙碌状态
** 函数说明 : 若12864 处于忙碌，等待直到空闲状态
** 入口参数 : 无
** 出口参数 : 无
*****************************************************************************
*********************/
extern void Check_12864_Busy(void);

/*****************************************************************************
***********************
** 函数名称 : Write_12864_Data(uchar date)
** 函数功能 : 写数据
** 函数说明 :
** 入口参数 : 待写入数据
** 出口参数 : 无
*****************************************************************************
*********************/
extern void Write_12864_Data(uchar date);//写数据

/*****************************************************************************
***********************
** 函数名称 : uchar Read_12864_Data(void)
** 函数功能 : 读数据
** 函数说明 :
```

```
** 入口参数 ：无
** 出口参数 ：待读出数据
*******************************************************************************
*********************/
extern uchar Read_12864_Data(void);//读数据

/*******************************************************************************
**********************
** 函数功能 ：读数据
** 函数说明 ：
** 入口参数 ：待写入的指令
** 出口参数 ：无
*******************************************************************************
*********************/
extern void Write_12864_Com(uchar commod);//写指令

/*******************************************************************************
**********************
** 函数功能 ：12864 液晶初始化程序
** 函数说明 ：
** 入口参数 ：无
** 出口参数 ：无
*******************************************************************************
*********************/
extern void LCD_12864_Init(void);


/*******************************************************************************
**********************
** 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
*******************************************************************************
*********************/
extern void Write_12864_String(uchar *str);//写入字符串或者汉字


/*******************************************************************************
**********************
** 函数功能 ：在指定位置写入汉字
** 函数说明 ：汉字必须是写在一个连续的 16*16 的点阵中
** 入口参数 ：待写入的汉字
** 函数举例 ：Hanzi_Disp("液晶实验")
** 出口参数 ：无
*******************************************************************************
*********************/
extern void Hanzi_Disp(uchar x,uchar y,uchar *s);

/*******************************************************************************
**********************
** 函数功能 ：在指定位置写一个整数
** 函数说明 ：整数的范围从 0 到 65535
** 入口参数 ：待写入的整数
** 出口参数 ：无
*******************************************************************************
*********************/
extern void Num_Display(uchar x,uchar y,uint number);

/*******************************************************************************
**********************
** 函数功能 ：在屏幕上显示一副图片
** 函数说明 ：可以有半屏显示和全屏显示两种模式，可以自行修改
** 画图方式为从左到右扫描，从上到下，每写一个字节后行地址加一，每写玩 16 个字节，即 128 个点
（如果半屏显示 8 个字节，64 个点）列地址加一
** 点阵取点方式为从左到右，从上到下，高位在前，低为在后
** 入口参数 ：绘画的图片的首地址
** 出口参数 ：无
*******************************************************************************
*********************/
extern void PHO_Display(const unsigned char *s);

/*******************************************************************************
**********************
** 函数功能 ：清除 12864 内部的 CGRAM
** 函数说明 ：当屏幕之前通过 CGRAM 绘图，可以调用此函数来清除 CGRAM 中的内容
** 入口参数 ：无
** 出口参数 ：无
*******************************************************************************
*********************/
extern void Clear_12864(void);


/*******************************************************************************
**********************
** 函数功能 ：在屏幕上打上一个点，屏幕像素是 128*64
** 函数说明 ：打点范围不要超过坐标范围，这个函数用来为后面的画直线和其他函数服务
** 入口参数 ：待写入的整数
** 出口参数 ：无
*******************************************************************************
*********************/
extern void Set_12864_Point(uchar x,uchar y);


/*******************************************************************************
**********************
** 函数名称 ：void LCD_12864_DrawLine(uint x1, uint y1, uint x2, uint y2)
** 函数功能 ：在 12864 屏幕上画一条直线
** 函数说明 ：因分辨率较低，斜线并不完美
```

```
** 入口参数 : x1:起点横坐标
                        y1:起点纵坐标
                        x2:终点横坐标
                        y2:终点纵坐标
*****************************************************************************
*********************/
extern void LCD_12864_DrawLine(uint x1, uint y1, uint x2, uint y2);


#endif

/*
void main()
{
        LCD_12864_Init();
        Clear_12864();
        PHO_Display(huashi);
        Write_12864_Com(0x84);
        Write_12864_String("Blue_Sky");
        Write_12864_Com(0x94);
        Write_12864_String(" I LIKE");
        Write_12864_Com(0x8d);
        Write_12864_String(" MCU");
        Write_12864_Com(0x9c);
        Write_12864_String("12 年03 月");
        while(1);
}
*/
```

lcd12864.c

```
/*****************************************************************************
**********************
** 文件功能 : 12864 液晶并行驱动程序
** 注意事项 : 硬件部分需要将12864 的第15 个管脚通过电阻接到高电平
** 工程作者 : Blue Sky Teams——ZZL
** 工程版本 : V1.0
*****************************************************************************
*********************/

 #include "parallel_12864.h"



uchar code addr_tab[32] = {
0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,//第一行汉字位置
0x90,0x91,0x92,0x93,0x94,0x95,0x96,0x97,//第二行汉字位置
0x88,0x89,0x8a,0x8b,0x8c,0x8d,0x8e,0x8f,//第三行汉字位置
0x98,0x99,0x9a,0x9b,0x9c,0x9d,0x9e,0x9f,//第四行汉字位置

};

/*****************************************************************************
**********************
** 函数名称 : void Check_12864_Busy(void)
** 函数功能 : 读忙碌状态
** 函数说明 : 若12864 处于忙碌，等待直到空闲状态
** 入口参数 : 无
** 出口参数 : 无
*****************************************************************************
*********************/
void Check_12864_Busy(void)
{
        uchar flag = 0 , i;
        DATA_PORT = 0XFF;
        RS = 0;
        RW = 1;
        while(flag == 0)
        {
                for(i = 0;i ++;i< 10)
                {
                        Delay_Us(1);
                }
                CE = 1;
                for(i = 0;i ++;i < 10)
                {
                        Delay_Us(1);
                }
                if(DATA_PORT & 0X80)flag = 0;
                else flag = 1;
                for(i = 0;i ++ ;i < 10)
                {
                        Delay_Us(1);
                }
                CE = 0;
        }
}
/*****************************************************************************
**********************
** 函数名称 : Write_12864_Data(uchar date)
** 函数功能 : 写数据
** 函数说明 :
** 入口参数 : 待写入数据
** 出口参数 : 无
*****************************************************************************
*********************/
void Write_12864_Data(uchar date)//写数据
{
        uchar i;
        Check_12864_Busy();
        RS = 1;
        RW = 0;
        for(i = 0;i ++;i < 4)
        {
                Delay_Us(1);
        }
        CE = 1;
```

```
        for(i = 0;i ++ ;i < 4)
        {
                Delay_Us(1);
        }
        DATA_PORT = date;
        for(i = 0;i ++;i < 4)
        {
                Delay_Us(1);
        }
        CE = 0;
}
/*****************************************************************************
**********************
** 函数名称 : uchar Read_12864_Data(void)
** 函数功能 : 读数据
** 函数说明 :
** 入口参数 : 无
** 出口参数 : 待读出数据
*****************************************************************************
*********************/
uchar Read_12864_Data(void)//读数据
{
        uchar date,i;
        Check_12864_Busy();
        DATA_PORT = 0XFF;
        RS = 1;
        Delay_Us(1);
        date = DATA_PORT;
        RW = 1;
        for(i = 0;i ++ ;i < 10)
        {
                Delay_Us(1);
        }
        CE = 1;
        for(i = 0;i ++;i < 10)
        {
                Delay_Us(1);
        }
        date = DATA_PORT;
        CE = 0;
        return(date);
}
/*****************************************************************************
**********************
** 函数功能 : 读数据
** 函数说明 :
** 入口参数 : 待写入的指令
** 出口参数 : 无
*****************************************************************************
*********************/
void Write_12864_Com(uchar commod)//写指令
{
        uchar i;
        Check_12864_Busy();
        RS = 0;
        RW = 0;
        for(i = 0;i ++;i < 4)
        {
                Delay_Us(1);
        }
        CE = 1;
        for(i = 0;i ++;i < 4)
        {
                Delay_Us(1);
        }
        DATA_PORT = commod;
        for(i = 0;i ++;i < 4)
        {
                Delay_Us(1);
        }
        CE = 0;
}
/*****************************************************************************
**********************
** 函数功能 : 12864 液晶初始化程序
** 函数说明 :
** 入口参数 : 无
** 出口参数 : 无
*****************************************************************************
*********************/

void LCD_12864_Init(void)
{
        PSB=1;              //设置为8BIT 并口工作模式
        RES=0;              //复位
        Delay_Us(200);
        RES=1;              //关复位
        Write_12864_Com(basic_commod);//30H--基本指令操作
        Write_12864_Com(cursor_off);//0x06 开显示，关光标，不闪烁。 扩展
        Write_12864_Com(clear);//清除显示0x01
        Write_12864_Com(0x06);//指定在资料写入或读取时，光标的移动方向，DDRAM 的地址计数
器（AC）加1。

}

/*****************************************************************************
**********************
** 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 Write_12864_String(uchar *str)//写入字符串或者汉字
{
        uchar *p;
        p = str;
        while(*p != 0)
        {
                Write_12864_Data(*p);
                p = ++str;
        }
}

/*******************************************************************************
************************
** 函数功能 : 在指定位置写入汉字
** 函数说明 : 汉字必须是写在一个连续的16*16 的点阵中
** 入口参数 : 待写入的汉字
** 函数举例 : Hanzi_Disp("液晶实验")
** 出口参数 : 无
*******************************************************************************
**********************/
void Hanzi_Disp(uchar x,uchar y,uchar *s)
{
        Write_12864_Com(addr_tab[8 * x + y]);  //写地址
        while(*s > 0)
           {
                        Write_12864_Data(*s);    //写数据
                        s ++;
           }
}
/*******************************************************************************
************************
** 函数功能 : 在指定位置写一个整数
** 函数说明 : 整数的范围从0 到65535
** 入口参数 : 待写入的整数
** 出口参数 : 无
*******************************************************************************
**********************/
void Num_Display(uchar x,uchar y,uint number)
{
        uint i = 10000;
        Write_12864_Com(addr_tab[8 * x + y]);  //写地址
        if( number !=0)
        {
                while(number / i == 0)
                {
                        i /= 10;
                }

                if(i != 1)
                {
                        while(number/i != 0)
                        {
                                Write_12864_Data(number / i + 0x30);
                                number -= (number / i) * i;
                                i /= 10;
                                if(i == 1)break;
                        }
                }
        }
        Write_12864_Data(number%10+0x30);
}
/*******************************************************************************
************************
** 函数功能 : 在屏幕上显示一副图片
** 函数说明 : 可以有半屏显示和全屏显示两种模式，可以自行修改
** 画图方式为从左到右扫描，从上到下，每写一个字节后行地址加一，每写玩16 个字节，即128 个点
（如果半屏显示8 个字节，64 个点）列地址加一
** 点阵取点方式为从左到右，从上到下，高位在前，低为在后
** 入口参数 : 绘画的图片的首地址
** 出口参数 : 无
*******************************************************************************
**********************/
void PHO_Display(const unsigned char *s)
{
        uchar COUNT3 = 0X02,COUNT1,COUNT2,LCD_X = 0X80,LCD_Y;

        for (;COUNT3 != 0;COUNT3 --)
        {
                LCD_Y = 0X80; //上半屏
                COUNT2 = 0X20;//32
                for(;COUNT2 != 0;COUNT2 --)
                {
        //                COUNT1 = 0X10;//8 个16*16 点阵单元 显示全屏
                        COUNT1 = 0X08;//4 个16*16 点阵单元 显示半屏 华师logo
                        Write_12864_Com(0x34);
                        Write_12864_Com(LCD_Y);
                        Write_12864_Com(LCD_X);
                        Write_12864_Com(0x30);
                        for (;COUNT1 != 0;COUNT1 --)
                        {
                                Write_12864_Data(*s ++);
                        }
                        LCD_Y += 1;
                }
                LCD_X = 0X88; //下半屏
        }
        Write_12864_Com(0x36);
        Write_12864_Com(0x30);
}
/*另外一种打点画图程序*/
void img_displeft(const unsigned char *img)  //  注意0---31，0---31 上下分半
{
    unsigned char i,j,m,n;
  for(j = 0;j < 32;j ++)
     {
```

```
      for(i = 0;i < 4;i ++)
      {
        Write_12864_Com(0x34); //扩展指令，显示绘图
        Write_12864_Com(0x80 + j); //更新坐标
        Write_12864_Com(0x80 + i);
        Write_12864_Com(0x30); //基本指令集
        Write_12864_Data(img[j * 8 + i *2 ]);  //高字节
        Write_12864_Data(img[j * 8 + i * 2 + 1]); //低字节
      }
    }
  for(n = 0;n < 32;n ++)
    {
      for(m = 0;m < 4;m ++)
      {
        Write_12864_Com(0x34); //扩展指令，显示绘图
        Write_12864_Com(0x80 + n); //更新坐标
        Write_12864_Com(0x88 + m);
        Write_12864_Com(0x30); //基本指令集
        Write_12864_Data(img[n * 8 + 256 + m * 2]);  //高字节
        Write_12864_Data(img[n * 8 + 256 + m * 2 + 1]); //低字节
      }
    }
    Write_12864_Com(0x36); //扩充功能指令，开绘图开关。
}
/*******************************************************************************
**********************
** 函数功能 ：清除12864内部的CGRAM
** 函数说明 ：当屏幕之前通过CGRAM绘图，可以调用此函数来清除CGRAM中的内容
** 入口参数 ：无
** 出口参数 ：无
*******************************************************************************
**********************/
void Clear_12864(void)
{
        uchar COUNT3 = 0X02,COUNT1,COUNT2,LCD_X = 0X80,LCD_Y;

        for (;COUNT3 != 0;COUNT3 --)
        {
                LCD_Y = 0X80;
                COUNT2 = 0X20;//32
                for(;COUNT2 != 0;COUNT2 --)
                {
                        COUNT1 = 0X10;//16
                        Write_12864_Com(0x34);
                        Write_12864_Com(LCD_Y);
                        Write_12864_Com(LCD_X);
                        Write_12864_Com(0x30);
                        for (;COUNT1 != 0;COUNT1 --)
                        {
                                Write_12864_Data(0x00);
                        }
                        LCD_Y += 1;
                }
                LCD_X = 0X88;
        }
        Write_12864_Com(0x36);
        Write_12864_Com(0x30);
}
/*******************************************************************************
**********************
** 函数功能 ：在屏幕上打上一个点，屏幕像素是128*64
** 函数说明 ：打点范围不要超过坐标范围，这个函数用来为后面的画直线和其他函数服务
** 入口参数 ：待写入的整数
** 出口参数 ：无
*******************************************************************************
**********************/
void Set_12864_Point(uchar x,uchar y)
{
        uchar x_byte,x_bit,y_byte,y_bit;
        uchar date1,date2;//存储读回的数据
        x &= 0x7F;//限制在液晶屏幕的范围内
        y &= 0x3F;

        x_byte = x / 16;//算出在那个字节
        x_bit = x & 0x0f;       //算出是哪一位

        y_byte = y / 32;
        y_bit = y & 0x3f;

        Write_12864_Com(extern_commod);
        Write_12864_Com(0x80 + y_bit);
        Write_12864_Com(0x80 + x_byte + 8 * y_byte);

        Read_12864_Data();//先空读一次？不知道为什么
        date1=Read_12864_Data();
        date2=Read_12864_Data();
        Write_12864_Com(extern_commod);
        Write_12864_Com(0x80 + y_bit);
        Write_12864_Com(0x80 + x_byte + 8 * y_byte);
        if(x_bit < 8)
        {
                Write_12864_Data(date1 | (0x01 << (7-x_bit) ) );
                Write_12864_Data(date2);
        }
        else
        {
                Write_12864_Data(date1);
                Write_12864_Data(date2 | (0x01 << (15-x_bit) ) );
        }
        Write_12864_Com(draw_on);
        Write_12864_Com(basic_commod);
}
/*******************************************************************************
**********************
** 函数名称 ：void LCD_12864_DrawLine(uint x1, uint y1, uint x2, uint y2)
** 函数功能 ：在12864屏幕上画一条直线
```

```
** 函数说明 ： 因分辨率较低，斜线并不完美
** 入口参数 ： x1:起点横坐标
                y1:起点纵坐标
                x2:终点横坐标
                y2:终点纵坐标
** 出口参数 ： 当前的键盘值
*******************************************************************************
*********************/
void LCD_12864_DrawLine(uint x1, uint y1, uint x2, uint y2)
{
        uint t;
        int xerr = 0,yerr = 0,delta_x,delta_y,distance;
        int incx,incy,uRow,uCol;

        delta_x = x2-x1; //计算坐标增量
        delta_y = y2-y1;
        uRow = x1;
        uCol = y1;
        if(delta_x > 0)incx = 1; //设置单步方向
        else if(delta_x == 0)incx = 0;//垂直线
        else {incx = -1;delta_x = - delta_x;}
        if(delta_y > 0)incy = 1;
        else if(delta_y == 0)incy = 0;//水平线
        else{incy = -1;delta_y = -delta_y;}
        if( delta_x > delta_y)distance = delta_x; //选取基本增量坐标轴
        else distance = delta_y;
        for(t = 0;t <= distance + 1;t ++ )//画线输出
        {
                Set_12864_Point(uRow,uCol);//画点
                xerr += delta_x ;
                yerr += delta_y ;
                if(xerr > distance)
                {
                        xerr -= distance;
                        uRow += incx;
                }
                if(yerr > distance)
                {
                        yerr -= distance;
                        uCol += incy;
                }
        }
}
```

15.pcf8591//AD 转换

pcf8591.h

```
/*******************************************************************************
*****************
工程名称：        AD_LED
功能描述：        向 EEPROM 连续存储多个字节数据，然后读取出来并在发光二极管上显示,实现流水
灯功能。
硬件连接：    用 8 位杜邦线将 J8 与 J12 连接，2 位杜邦线分别将 P2.0 与 J15_DS1、P2.1 与 J15_DS2
连接
维护记录：    2011-8-22
*******************************************************************************
***************/
#ifndef _PCF8591_H_
#define _PCF8591_H_

  #ifndef uchar
        #define uchar unsigned char
  #endif

  #ifndef uint
        #define uint  unsigned int
  #endif

#include"intrins.h"
#include "STC12C5A60S2.H"

sbit PCF8591_sda=P2^7;          //数据线
sbit PCF8591_scl=P2^6;          //时钟线


 /*******************************************************************************
***********************
** 函数功能 ： 延时函数
** 函数说明 ： 利用软件延时，占用 CPU，经调试最小单位大约为 1us
** 入口参数 ： time:需要延时的时间，单位 us
** 出口参数 ： 无
*******************************************************************************
*********************/
extern void Delay_Us(uchar n);


//*****************************************************************************
*****************
//启动(SCL 为高，SDA 由高变为低是一个开始条件)
//*****************************************************************************
*****************
extern void PCF8591_start();

//*****************************************************************************
*****************
//停止（SCL 为高，SDA 由低变为高是一个结束条件）
//*****************************************************************************
*****************
extern void PCF8591_stop();

//*****************************************************************************
*****************
//检测应答(所有的地址和数据字都是以 8bit,在第 9 个时钟周期，从器件发出"0"信号来作为收到一个
字的应答信号)
```

```
//*****************************************************************************
*****************
extern void PCF8591_checkACK();                    //主器件检测从器件是否返回应答

//*****************************************************************************
*****************
//发送应答(发送方为主器件，接收方为从器件，控制器作为从器件接收完1 数据时，发送应答信号
//*****************************************************************************
*****************
extern void PCF8591_sendACK(bit ACK);


//*****************************************************************************
*****************
//写一字节
//*****************************************************************************
*****************
extern void PCF8591_send_byte(uchar date);       //写一个8 位字


//*****************************************************************************
*****************
//读一字节
//*****************************************************************************
*****************
extern uchar PCF8591_receive_byte();        //读一个8 位字



//*****************************************************************************
*****************
//读取AD 转换结果数据
//*****************************************************************************
*****************
extern uchar PCF8591_Read_AD(uchar chn);


//*****************************************************************************
*****************
//DA 转化函数
//*****************************************************************************
*****************
extern void PCF8591_DAC(uchar temp);



#endif
```

pcf8591.c

```
/*****************************************************************************
*****************
维护记录：   2011-8-22
*****************************************************************************
***************/

#include "PCF8591.h"


//*****************************************************************************
*****************
//启动(SCL 为高，SDA 由高变为低是一个开始条件)
//*****************************************************************************
*****************
void PCF8591_start()
{
        PCF8591_sda=1; _nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop
_();   //数据线置高,
        PCF8591_scl=1; _nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop
_();   //时钟线置高
        PCF8591_sda=0; _nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop
_();   //数据线置低,由高变低
        PCF8591_scl=0; _nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop
_();   //时钟线置低,准备发送或接收数据,总线进入忙状态(I2C 总线在空闲状态时，SDA 与SCL 均被
置高)
        Delay_Us(1);  //延时
}
//*****************************************************************************
*****************
//停止（SCL 为高，SDA 由低变为高是一个结束条件）
//*****************************************************************************
*****************
void PCF8591_stop()
{
        PCF8591_sda=0;                  //数据线置低
        Delay_Us(1);                 //延时
        PCF8591_scl=1;                  //时钟线置高
        Delay_Us(1);                 //延时
        PCF8591_sda=1;                  //数据线置高，由低变高
        Delay_Us(1);                 //延时
}
//*****************************************************************************
*****************
//检测应答(所有的地址和数据字都是以8bit,在第9 个时钟周期，从器件发出"0"信号来作为收到一个
字的应答信号)
//*****************************************************************************
*****************
void PCF8591_checkACK()                    //主器件检测从器件是否返回应答
{
        PCF8591_scl=1;                      //时钟线置高
        Delay_Us(1);                       //延时
        while(PCF8591_sda==1);                  //等待第9 个时钟周期器件发出的响应信号"0
"
        PCF8591_scl=0;                      //时钟线置低
```

```
        Delay_Us(1);                        //延时
}
//*****************************************************************************
*****************
//发送应答(发送方为主器件，接收方为从器件，控制器作为从器件接收完1 数据时，发送应答信号
//*****************************************************************************
*****************
void PCF8591_sendACK(bit ACK)
{
    if(ACK) PCF8591_sda=1;            //如果i 位为1 则发送1,即发送"非应答信号"
      else PCF8591_sda=0;             //如果i 位为0 则发送0,即发送"应答信号"
    PCF8591_scl=1;                    //时钟线置高,给一个脉冲
        Delay_Us(1);                  //延时
        PCF8591_scl=0;                    //时钟线置低
        Delay_Us(1);                  //延时
}
//*****************************************************************************
*****************
//写一字节
//*****************************************************************************
*****************
void PCF8591_send_byte(uchar date)       //写一个8 位字
{
        uchar i,temp;                 //定义局部变量
        temp=date;                    //待发8 位数据赋予temp
        for(i=0;i<8;i++)              //循环8 次，每次写入1 位，从最高位开始发送
        {
        if(temp&0x80) PCF8591_sda=1;      //如果temp 最高位为1 则发送1
          else   PCF8591_sda=0;           //如果temp 最高位为0 则发送0
            _nop_();                  //延时
                PCF8591_scl=1;                    //给一个脉冲，发送sda 当前这位数据
                Delay_Us(5);                  //延时,需大于4us(参考数据手册时序图)

                PCF8591_scl=0;                    //时钟线置低,准备下一脉冲
            Delay_Us(6);                  //延时,需大于4.7us(参考数据手册时序图)

                temp=temp<<1;             //左移1 位，准备好下1 位待发送的数据
        }
        PCF8591_checkACK();                   //查询是否返回应答信号
}
//*****************************************************************************
*****************
//读一字节
//*****************************************************************************
*****************
 uchar PCF8591_receive_byte()         //读一个8 位字
{
        uchar i,temp;             //定义局部变量
        PCF8591_sda=1;                    //设置数据线为输入
        _nop_();                  //延时
        for(i=0;i<8;i++)          //循环8 次，每次读取1 位，从最高位开始接收
        {
                PCF8591_scl=1;                //给一脉冲，准备发送1 位数据

                Delay_Us(5);              //延时,需大于4us(参考数据手册时序图)

                temp=(temp<<1)|PCF8591_sda;  //读取1 位数据,放在temp 最低位
                PCF8591_scl=0;                //准备给下1 个脉冲
            Delay_Us(6);              //延时,需大于4.7us(参考数据手册时序图)

        }
    PCF8591_sendACK(0);               //发送应答信号
        return temp;              //返回读取的8 位数据
}
//*****************************************************************************
*****************
//读取AD 转换结果数据
//*****************************************************************************
*****************
uchar PCF8591_Read_AD(uchar chn)
 {
   uchar  ad_data;           //定义变量，存放转换结果
   PCF8591_stop();                   //停止
   PCF8591_start();                  //启动总线
   PCF8591_send_byte(0x90);          //选择从器件地址,RW 位为0，即选择写命令
   PCF8591_send_byte(0x40|chn);      //寄存器设置，0 通道

   PCF8591_start();                  //启动I2C 总线
   PCF8591_send_byte(0x91);          //选择从器件地址,RW 位为1，即选择读命令

   ad_data=PCF8591_receive_byte();  //读取转换结果
   PCF8591_sendACK(1);               //发送非应答信号
   PCF8591_stop();                   //停止
   return(ad_data);
 }

//*****************************************************************************
*****************
//DA 转化函数
//*****************************************************************************
*****************
void PCF8591_DAC(uchar v)
 {
   PCF8591_stop();                   //停止
   PCF8591_start();                  //启动总线
   PCF8591_send_byte(0x90);          //选择从器件地址,RW 位为0，即选择写命令
   PCF8591_send_byte(0x40);          //寄存器设置
   PCF8591_send_byte(v);         //写入数字量
   PCF8591_stop();                   //停止
 }
```

16.PWM//直流电机

pwm.h

```
/*********************************************************************************
**********************
** 工程功能 : PWM 输出
** 工程作者 : Blue Sky Teams——WCW
** 工程版本 : V1.0
*********************************************************************************
*********************/

#ifndef _PWM_H_
#define _PWM_H_

#include "STC12C5A60S2.H"      //包含头文件

  #ifndef uchar
        #define uchar unsigned char
  #endif

  #ifndef uint
        #define uint  unsigned int
  #endif


extern void PCA_init(void);


#endif
```

pwm.c

```
/*********************************************************************************
**********************
** 工程功能 : PWM 输出
** 工程作者 : Blue Sky Teams——WCW
** 工程版本 : V1.0
*********************************************************************************
*********************/

#include "PWM.h"

//详细的介绍请参看 STC12C5A60S2.pdf 中第 10 章 STC12C5A60S2 系列单片机 PCA/PWM 的应用

#define thx0 0xFF
#define tlx0 0xFF
/*********************************************************************************
**********************
** 函数功能 : 初始化 PCA
** 函数说明 : 把 PCA 的时钟源设置为 timer0 的溢出频率;
** 入口参数 : 无
** 出口参数 : 无
*********************************************************************************
*********************/
void PCA_init()
{
        CMOD=0X80;       //CIDL=1;在空闲模式下停止 PCA 计数器工作;
                         //0X84   时钟源设置为 timer0 的溢出频率;
                         //0x80   系统时钟 12 分频
                         //ECF=0;禁止 PCA 计数溢出中断功能;
        CCON=0X00;       //CF=0;清零 PCA 溢出中断请求标志位;
                         //CR=0;不允许 PCA 计数器计数
                         //清零 PCA 各模块中断请求标志位
        CH=0;            //清零 PCA 计数器
        CL=0;

       CCAPM0=0x42; //设置模块 0 为 8 位 PWM 输出模式;PWM 无需中断支持;模块 0 的脉冲在 P1.
3 脚输出
        PCA_PWM0=0x00;
        CR=1;             //启动 PCA 计数器

        //通过控制 CCAPnH 来控制占空比                        CCAP0H   小于 CCAP0H 输出 0,
大于时输出 1
        //CL 跳变到 00 时,CCAPnH 自动装载到 CCAPnL
}

///*********************************************************************************
************************
//** 函数功能 : 初始化 Timer0
//** 函数说明 : Timer0 的溢出率为 PCA 计数器的时钟
//** 入口参数 : 无
//** 出口参数 : 无
//*********************************************************************************
************************/
//void Timer0_init()
//{
//      TMOD=0x02;                          //计数器 0 设置为 8 位计数器且自动重载
//      TH0=thx0;         //十个系统时钟 Timer0 溢出一次
//      TL0=tlx0;
//      TR0=1;                              //开定时器 0
//}
```

17.stepper_motor//步进电机

stepper_motor.h

```
/*********************************************************************************
****
*                                步进电机实验
                                                                    *
实现现象：下载程序后，按照光盘内操作视频接线，步进电机旋转
注意事项：无
```

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

#ifndef _STEPPER_MOTOR_H_
#define _STEPPER_MOTOR_H_

#include "STC12C5A60S2.H"      //包含头文件

  #ifndef uchar
        #define uchar unsigned char
  #endif

  #ifndef uint
        #define uint  unsigned int
  #endif

sbit MOTOA = P1^0;
sbit MOTOB = P1^1;
sbit MOTOC = P1^2;
sbit MOTOD = P1^3;


extern void stepper_motor(uchar Dir);


#endif
```

stepper_motor.c

```
#include "stepper_motor.h"

/*******************************************************************************
* 输    入       : Dir 1:正转   0:倒转  >2:停转
* 输    出        : 无
 ，每执行一次转动一定角度，执行间隔时间决定速度，可放在定时中断，改变中断时间
*******************************************************************************/
void stepper_motor(uchar Dir)
{
        static char phase=0;
        if(Dir<2){
                switch(phase)
                {
                case(0):
                        MOTOA = 1;
                        MOTOB = 0;
                        MOTOC = 0;
                        MOTOD = 0;
                        break;

                case(1):
                        MOTOA = 0;
                        MOTOB = 1;
                        MOTOC = 0;
                        MOTOD = 0;
                        break;

                case(2):
                        MOTOA = 0;
                        MOTOB = 0;
                        MOTOC = 1;
                        MOTOD = 0;
                        break;

                case(3):
                        MOTOA = 0;
                        MOTOB = 0;
                        MOTOC = 0;
                        MOTOD = 1;
                        break;
                }
        }else{//停转
                        MOTOA = 0;
                        MOTOB = 0;
                        MOTOC = 0;
                        MOTOD = 0;

        }
        if(Dir==1)
                phase++;
        if(Dir==0)
                phase--;

        if(phase>3)
                phase=0;
        if(phase<0)
                phase=3;
}
```

18.timer 定时器内部中断

timer.h

```
/*******************************************************************************************
*****************
工程名称:        timer
功能描述:        定时器0 实现1s 定时，流水灯显示上的数据每秒加1
硬件连接:    用8 位杜邦线将J8 与J13 连接
维护记录:    2011-8-22
*******************************************************************************************
***************/


#ifndef _TIME_H_
```

```
#define _TIME_H_

#include "STC12C5A60S2.H"      //包含头文件

  #ifndef uchar
        #define uchar unsigned char
  #endif

  #ifndef uint
        #define uint  unsigned int
  #endif
//*************************************************************************
//定时器0 初始化    16 位 10ms 中断    可作为PCA 时钟源（PWM）
//*************************************************************************
extern void Timer0_init(void);

//*************************************************************************
//定时器1 初始化  16 位定时器 1ms 中断 //优先级3(最高)
//*************************************************************************
extern void Timer1_Init(void);


#endif
```

timer.c

```
/*************************************************************************************
*****************
工程名称:         timer
功能描述:         定时器0 实现1s 定时，流水灯显示上的数据每秒加1
硬件连接:    用8 位杜邦线将J8 与J13 连接
维护记录:    2011-8-22
*************************************************************************************
***************/
#include "time.h"
//*************************************************************************
//定时器0 初始化    16 位 10ms 中断    可作为PCA 时钟源（PWM）
//*************************************************************************
void Timer0_init(void)
{
        //AUXR = AUXR|0x80;  // T0, 1T Mode
        TMOD &= 0xF0;
        TMOD |= 0x01;
        TH0  = (65535-10000)/256;   //10ms
        TL0  = (65535-10000)%256;
        ET1  = 1;  //开启中断
        TR0  = 1;
}

//*************************************************************************
//定时器1 初始化  16 位定时器 1ms 中断 //优先级3(最高)
//*************************************************************************
void Timer1_Init(void)
{
        TMOD &= 0x0F;
    TMOD |= 0x10;
        //AUXR = AUXR|0x40;  // T1, 1T Mode
        TH0  = (65535-1000)/256;   //1ms
        TL0  = (65535-1000)%256;
        //IPH |= 1<<3; //优先级设置
        //IP  |= 1<<3; //00 最低，11 最高
    EA = 1;
    ET1 = 1;
    TR1 = 1;
}

//void main(void)
//{
//    InitTimer1();
//}

void Timer0Interrupt(void) interrupt 1
{
        TH0  = (65535-10000)/256;   //10ms
        TL0  = (65535-10000)%256;
    //add your code here!
}

void Timer1Interrupt(void) interrupt 3
{
        TH0  = (65535-1000)/256;   //1ms
        TL0  = (65535-1000)%256;
    //add your code here!
}
```

19.usart 串口

usart.h

```
/*************************************************************************************
**********************
** 文件功能 : 串口通信相关程序
** 工程作者 : Blue Sky Teams——ZZL
** 工程版本 : V1.0
*************************************************************************************
**********************/


#ifndef _USART_H_
#define _USART_H_

#include "STC12C5A60S2.H"      //包含头文件

  #ifndef uchar
        #define uchar unsigned char
  #endif
```

```
  #ifndef uint
        #define uint  unsigned int
  #endif


extern  uchar data_length,flag; //声明外部变量
extern  uchar data_buffer[10];  //缓存区

/*******************************************************************************
**********************
** 函数功能 : 串口初始化
** 函数说明 : SMOD 是电源控制寄存器中 PCON 的 bit7 位
**T1x12 是辅助寄存器 AUXR 中的 bit6 位
** 实现串口外设的配置，波特率计算方法：SMOD   T1x12
**                                    0       0       FSCLK/32/12/(256-TH1)
**                                    0       1       FSCLK/32/(256-TH1)
**                                    1       0       FSCLK*2/32/12(256-TH1)
**                                    1       1       FSCKL*2/32/(256-TH1)
** 入口参数 : 无
** 出口参数 : 无
** 详细的介绍请参看 STC12C5A60S2.pdf 第八章 串行口通信
*******************************************************************************
**********************/
extern void USART_Init();

/*******************************************************************************
**********************
** 函数功能 : 发送一个字符
** 入口参数 : d：待发送的字符
** 出口参数 : 无
*******************************************************************************
**********************/
extern void Send_Byte(unsigned char c);

/*******************************************************************************
**********************
** 函数功能 : 发送一定长度的字符串
** 入口参数 : str：待发送字符串的地址
**            length：发送字符串的长度
** 出口参数 : 无
*******************************************************************************
**********************/
extern void Send_String(uchar *str ,uint length);

unsigned char Uart_Receive_Byte(void);

/************************ (C) COPYRIGHT 2011 Blue Sky Teams *****END OF FILE*****
************************/


#endif
```

usart.c

```
/*******************************************************************************
**********************
** 文件功能 : 串口通信相关程序
** 工程作者 : Blue Sky Teams——ZZL
** 工程版本 : V1.0
*******************************************************************************
**********************/


#include "usart.h"

uchar data_length=0,flag=0; //声明外部变量
uchar data_buffer[10]={0};  //缓存区

/*******************************************************************************
**********************
** 函数功能 : 串口初始化
** 函数说明 : SMOD 是电源控制寄存器中 PCON 的 bit7 位
**T1x12 是辅助寄存器 AUXR 中的 bit6 位
** 实现串口外设的配置，波特率计算方法：SMOD   T1x12
**                                    0       0       FSCLK/32/12/(256-TH1)
**                                    0       1       FSCLK/32/(256-TH1)
**                                    1       0       FSCLK*2/32/12(256-TH1)
**                                    1       1       FSCKL*2/32/(256-TH1)
** 入口参数 : 无
** 出口参数 : 无
** 详细的介绍请参看 STC12C5A60S2.pdf 第八章 串行口通信
*******************************************************************************
**********************/
void USART_Init()
{
        PCON |= 0x80;           //使能波特率倍速位 SMOD
        SCON = 0x50;            //8 位数据,可变波特率
//      AUXR &= 0xBF;           //定时器 1 时钟为 Fosc/12,即 12T
//      AUXR &= 0xFE;           //串口 1 选择定时器 1 为波特率发生器
//      TMOD &= 0x0F;           //清除定时器 1 模式位
        TMOD |= 0x20;           //设定定时器 1 为 8 位自动重装方式
        TL1 = 0xFF;             //设定定时初值
        TH1 = 0xFF;             //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
        EA=1;

//      PCON &= 0x7F;           //波特率不倍速 SMOD=0
//      SCON = 0x50;            //设置为方式 1,8 位数据,可变波特率,接收允许
//      AUXR |= 0x40;           //定时器 1 时钟为 Fosc,即 1T
//      TMOD = 0X20;            //定时器 1:模式 2,8 位自动重装模式,用于产生波特率
//      TL1 = 0XD9;                 //设定定时初值，波特率设置为 9600
//      TH1 = 0XD9;                 //设定定时器重装值
//      TR1 = 1;                //启动定时器 1
```

```
//      ES = 1;                         //开串行中断
//      EA = 1;                         //开总中断
        //EA、ES 置1 后，若有串口接收或者发送，则进入执行串口中断服务程序void USART_Inte
rrupt(void)  interrupt 4
}
/*******************************************************************************
**********************
** 函数功能 : 发送一个字符
** 入口参数 : d：待发送的字符
** 出口参数 : 无
*******************************************************************************
**********************/
void Send_Byte(unsigned char c)
{

        SBUF = c;
        while(!TI);     //若发送完成，则TI 自动置1，则跳出循环，执行接下来的语句
        TI=0;           // 发送完成,TI 必须软件置零，下次发送的时候才硬件才可以将它置一
作为判断发送完成的依据
}
/*******************************************************************************
**********************
** 函数功能 : 发送一定长度的字符串
** 入口参数 : str：待发送字符串的地址
**            length：发送字符串的长度
** 出口参数 : 无
*******************************************************************************
**********************/
void Send_String(uchar *str ,uint length)
{
        while(length!=0)
        {
                Send_Byte(*str++) ;
                length --;
        }
}

/*******************************************************************************
**********************
** 函数功能 : 串口中断服务程序
** 入口参数 : 无
** 出口参数 : 无
*******************************************************************************
**********************/
void USART_Interrupt(void)  interrupt 4
{
        EA = 0;
        if(RI)                  //必须判断RI(接收中断标志)是否为1
    {
        RI=0;
        data_buffer[data_length++] = SBUF;//将接收到的数据存入到缓冲区内
               if(data_buffer[data_length-1] == '\n' || data_length == 10)flag = 1;
 //若受到回车键或缓冲区数据接收已满，置位flag
    }
        EA = 1;
}


unsigned char Uart_Receive_Byte()//UART Receive a byteg
{
        unsigned char dat;
        while(!RI);      //接收完为1
        RI = 0;
        dat = SBUF;
        return (dat);
}
/************************** (C) COPYRIGHT 2011 Blue Sky Teams *****END OF FILE*****
**********************/


//RS485 通讯实现

/*
功能描述：        实现485 双向通信
硬件连接：   用3 位杜邦线分别将J9_0 与J17_RO、J9_1 与J17_DI 以及J9_2 与J17_RE 连接,
             用1 位杜邦线将J11_0 与J7_S17 连接,用1 位杜邦线将J10_0 与J13_8 连接,
             将甲板和乙板RS485 接口的A、B 对应（即A 对A，B 对B）连接。
*/

//sbit    RO     =P3^0;   //定义RO
//sbit    DI     =P3^1;   //定义DI
//sbit    RE     =P3^2;   //定义RE,用于发送接收模式选择
//
//void main(void)
//{
// init_com( ) ;        //初始化串口
// while(1)
// {
//  if(KEY ==0 )        //判断是否有按键按下
//   {
//    delay(100);       //延时消抖
//    while(KEY ==0);   //等待按键释放
//    RE = 1;           //设置485 为发送模式
//    send_char(0xaa);  //发送1 字节数据
//   }
//
//  RE=0;               //设置485 为接收模式
//  if (read_flag)      //如果取数标志已置位，表示有接收到数据
//    {
//      read_flag= 0 ;  //取数标志清0
//      if(ch==0xaa)    //判断485 接收到的数据是否正确
//       {LED=0;delay(5000);LED=1;} //如果正确，LED 闪烁1 下
//    }
// }
//}
```