**江苏科技大学**

**单片机课程设计报告**

学 院 电子信息学院

专 业 电子信息工程

学生姓名 郑中权 王园园（24）

班级学号 152210303127 152210303208

指导教师 李彦

2018年 6 月

**目录**

**[第一章 课题背景](#_Toc573_WPSOffice_Level1)**[1](#_Toc573_WPSOffice_Level1)

**[第二章 基本要求](#_Toc17555_WPSOffice_Level1)**[2](#_Toc17555_WPSOffice_Level1)

**[第三章 系统设计](#_Toc2195_WPSOffice_Level1)**[3](#_Toc2195_WPSOffice_Level1)

[3.1硬件设计](#_Toc17555_WPSOffice_Level2) [3](#_Toc17555_WPSOffice_Level2)

[3.1.1控制模块](#_Toc2195_WPSOffice_Level3) [4](#_Toc2195_WPSOffice_Level3)

[3.1.2温度检测模块](#_Toc1418_WPSOffice_Level3) [6](#_Toc1418_WPSOffice_Level3)

[3.1.3显示模块](#_Toc7391_WPSOffice_Level3) [8](#_Toc7391_WPSOffice_Level3)

[3.1.4 串口通信](#_Toc29183_WPSOffice_Level3) [12](#_Toc29183_WPSOffice_Level3)

[3.1.5 报警模块](#_Toc5105_WPSOffice_Level3) [14](#_Toc5105_WPSOffice_Level3)

[3.1.6 按键模块](#_Toc20729_WPSOffice_Level3) [15](#_Toc20729_WPSOffice_Level3)

[3.2 软件设计](#_Toc2195_WPSOffice_Level2) [16](#_Toc2195_WPSOffice_Level2)

[3.2.1流程图](#_Toc1057_WPSOffice_Level3) [16](#_Toc1057_WPSOffice_Level3)

[3.2.2系统特点](#_Toc19141_WPSOffice_Level3) [19](#_Toc19141_WPSOffice_Level3)

3.2.3程序设计 21

**结论**-------------------------------------------------------------34

**参考文献**-------------------------------------------------------35

**第一章 课题背景**

温度是一个非常重要的物理量，因为它直接影响燃烧、化学反应、发酵、烘烤、煅烧、蒸馏、浓度、挤压成型、结晶及空气流动我物理和化学过程。对温度的采集监控和警报对保证生产的自动化、智能化能顺利、安全地进行，提高企业的生产效率有重要意义。温度采集控制系统在工业生产、科学研究和生活领域得到了广泛的应用，尤其是小型无人值守场所的温度测量，控制和警报，例如机房、实验室、大棚种植、地窖储藏等，同时也是冷链的一种重要环节，在食品储藏和运输，有医用冰箱贮存试剂、血液等也有重要的作用。温度控制失误可能引起生产安全、产品质量等一系列问题，严重的可能会带来人身安全和社会影响。

国外对温度监控技术的研究较早，始于20世纪70年代。显示采用采用模拟式的组合仪表采集现场信息并进行标记、记录和控制。80年代末出现了分布式控制系统。而我国对于温度监控系统的研究教育始于20世纪80年代，我国技术人员在吸收发达国家温度监测技术的基础上才慢慢掌握了温度室内的微型计算机控制技术。不同的生产情况和场合，控制方案不同，如PID控制、直接数字控制、推断控制等。但传统的控制方式已经不能再满足高精度，高速度的控制要求。采用MSC-51单片机对温度进行控制，不仅具有体积小、功能强、控制方便、组态简单和灵活性大等优点，而且可以大幅度提高被控温度的技术指标，从而提高产品的质量和数量。

**第二章 基本要求**

本设计要求利用PROTEUS仿真软件设计一个温度监控系统。

①以单片机AT89C51为核心，应用DS18B20终端来模拟温度传感器得到当前环境温度，通过单片机读取数据并将温度实时显示在LCD1602液晶显示屏上。

②要求系统能够自由设置温度报警上下限，并能够通过液晶屏显示出温度上限和下限数值。

③监测到当前环境温度高于设置的温度上限时，高温报警灯亮，电机正转；监测到当前环境温度低于设置的温度下限时，低温报警灯亮，电机反转；正常情况下，报警灯熄灭，电机停转。

④要求实现多单片机通信。

**第三章 系统设计**

**3.1硬件设计**

系统包含两个AT89C51芯片。芯片1连接3个按键，1个DS18B20温度传感器，1个LCD1602显示屏。芯片2连接2个LED灯，1个电机。

用AT89C51 单片机通过DS18B20监测当前温度，并将当前温度显示在 LCD1602上，单片机预设最高温度和最低温度。

通过按键1来设置，按一下开始设置最高温度，再按一下开始设置最低温度，按第三次返回到当前温度显示，来回反复。按键2设置温度增加，按键3设置温度减小。

当当前温度超过最高温度时，LED1闪烁，电机正转；低于最低温度时，LED2 闪烁，电机反转。温度正常时，LED1和LED2 熄灭，电机静止。

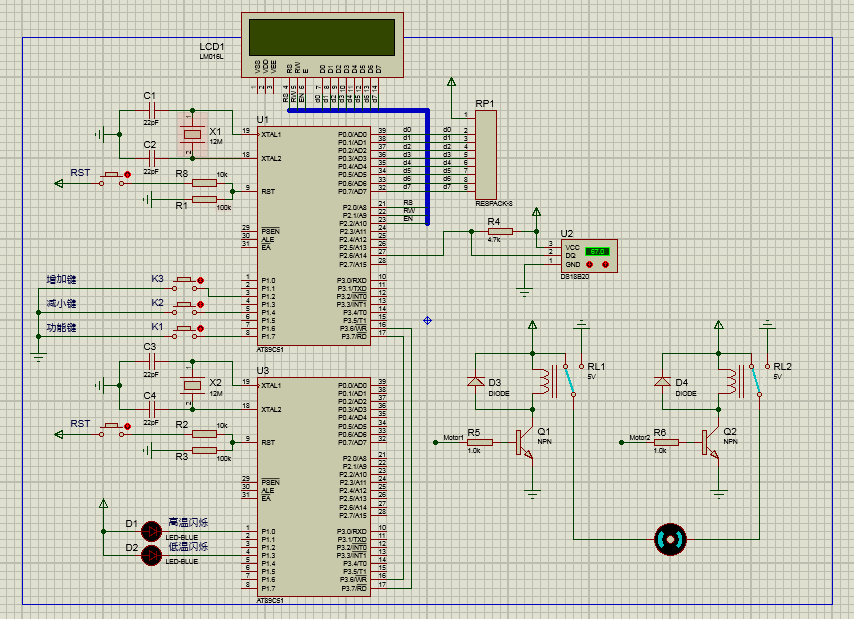


图3-1 系统整体设计电路图

**3.1.1控制模块**

单片机AT89C51

(1)简介

AT89C51是美国ATMEL公司生产的低电压，高性能CMOS8位单片机，片内含4k bytes的可反复擦写的只读程序存储器（PEROM）和128 bytes的随机存取数据存储器（RAM），器件采用ATMEL公司的高密度、非易失性存储技术生产，兼容标准MCS-51指令系统，片内置通用8位中央处理器（CPU）和Flash存储单元，功能强大AT89C51单片机可为您提供许多高性价比的应用场合，可灵活应用于各种控制领域。

(2)主要功能

AT89C51 提供以下标准功能：4k字节Flash闪速存储器，128字节内部RAM，32 个I/O 口线，两个16位定时/计数器，一个5向量两级中断结构，一个全双工串行通信口，片内振荡器及时钟电路。同时，AT89C51可降至0Hz的静态逻辑操作，并支持两种软件可选的节电工作模式。空闲方式停止CPU的工作，但允许RAM，定时/计数器，串行通信口及中断系统继续工作。掉电方式保存RAM中的内容，但振荡器停止工作并禁止其它所有部件工作直到下一个硬件复位。

**引脚功能：**

**VCC**：供电电压。

**GND**：接地。

**P0口**：P0口为一个8位漏级开路双向I/O口，每脚可吸收8TTL门电流。当P0口的管脚第一次写1时，被定义为高阻输入。P0能够用于外部程序数据存储器，它可以被定义为数据/地址的低八位。在FIASH编程时，P0 口作为原码输入口，当FIASH进行校验时，P0输出原码，此时P0外部必须接上拉电阻。

**P1口**：P1口是一个内部提供上拉电阻的8位双向I/O口，P1口缓冲器能接收输出4TTL门电流。P1口管脚写入1后，被内部上拉为高，可用作输入，P1口被外部下拉为低电平时，将输出电流，这是由于内部上拉的缘故。在FLASH编程和校验时，P1口作为低八位地址接收。

**P2口**：P2口为一个内部上拉电阻的8位双向I/O口，P2口缓冲器可接收，输出4个TTL门电流，当P2口被写“1”时，其管脚被内部上拉电阻拉高，且作为输入。并因此作为输入时，P2口的管脚被外部拉低，将输出电流。这是由于内部上拉的缘故。P2口当用于外部程序存储器或16位地址外部数据存储器进行存取时，P2口输出地址的高八位。在给出地址“1”时，它利用内部上拉优势，当对外部八位地址数据存储器进行读写时，P2口输出其特殊功能寄存器的内容。P2口在FLASH编程和校验时接收高八位地址信号和控制信号。

**P3口**：P3口管脚是8个带内部上拉电阻的双向I/O口，可接收输出4个TTL门电流。当P3口写入“1”后，它们被内部上拉为高电平，并用作输入。作为输入，由于外部下拉为低电平，P3口将输出电流（ILL）这是由于上拉的缘故。

P3口也可作为AT89C51的一些特殊功能口，如下表所示：

口管脚 备选功能

P3.0 RXD（串行输入口）

P3.1 TXD（串行输出口）

P3.2 /INT0（外部中断0）

P3.3 /INT1（外部中断1）

P3.4 T0（计时器0外部输入）

P3.5 T1（计时器1外部输入）

P3.6 /WR（外部数据存储器写选通）

P3.7 /RD（外部数据存储器读选通）

P3口同时为闪烁编程和编程校验接收一些控制信号。

**RST**：复位输入。当振荡器复位器件时，要保持RST脚两个机器周期的高电平时间。

**ALE/PROG**：当访问外部存储器时，地址锁存允许的输出电平用于锁存地址的低位字节。在FLASH编程期间，此引脚用于输入编程脉冲。在平时，ALE端以不变的频率周期输出正脉冲信号，此频率为振荡器频率的1/6。因此它可用作对外部输出的脉冲或用于定时目的。然而要注意的是：每当用作外部数据存储器时，将跳过一个ALE脉冲。如想禁止ALE的输出可在SFR8EH地址上置0。此时， ALE只有在执行MOVX，MOVC指令是ALE才起作用。另外，该引脚被略微拉高。如果微处理器在外部执行状态ALE禁止，置位无效。

**/PSEN**：外部程序存储器的选通信号。在由外部程序存储器取指期间，每个机器周期两次/PSEN有效。但在访问外部数据存储器时，这两次有效的/PSEN信号将不出现。

**/EA/VPP**：当/EA保持低电平时，则在此期间外部程序存储器（0000H-FFFFH），不管是否有内部程序存储器。注意加密方式1时，/EA将内部锁定为RESET；当/EA端保持高电平时，此间内部程序存储器。在FLASH编程期间，此引脚也用于施加12V编程电源（VPP）。

**XTAL1**：反向振荡放大器的输入及内部时钟工作电路的输入。

**XTAL2**：来自反向振荡器的输出。

**振荡器特性:**

XTAL1和XTAL2分别为反向放大器的输入和输出。该反向放大器可以配置为片内振荡器。石晶振荡和陶瓷振荡均可采用。如采用外部时钟源驱动器件，XTAL2应不接。有余输入至内部时钟信号要通过一个二分频触发器，因此对外部时钟信号的脉宽无任何要求，但必须保证脉冲的高低电平要求的宽度。

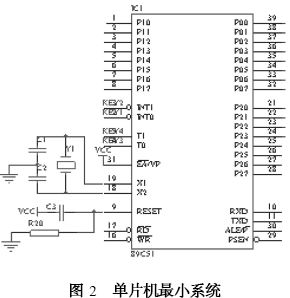


图3-2 单片机最小系统

**3.1.2温度检测模块**

DS18B20温度传感器

⑴ 主要特性：

①适应电压范围更宽，电压范围：3.0～5.5V，在寄生电源方式下可由数据线供电  
　　②独特的单线接口方式，DS18B20在与微处理器连接时仅需要一条口线即可实现微处理器与DS18B20的双向通讯  
　　③DS18B20支持多点组网功能，多个DS18B20可以并联在唯一的三线上，实现组网多点测温  
　　④DS18B20在使用中不需要任何外围元件，全部传感元件及转换电路集成在形如一只三极管的集成电路内  
　　⑤温度范围－55℃～＋125℃，在-10～+85℃时精度为±0.5℃  
　　⑥可编程的分辨率为9～12位，对应的可分辨温度分别为0.5℃、0.25℃、0.125℃和0.0625℃，可实现高精度测温  
　　⑦在9位分辨率时最多在93.75ms内把温度转换为数字，12位分辨率时最多在750ms内把温度值转换为数字，速度更快  
　　⑧测量结果直接输出数字温度信号，以"一线总线"串行传送给CPU，同时可传送CRC校验码，具有极强的抗干扰纠错能力  
　　⑨负压特性：电源极性接反时，芯片不会因发热而烧毁，但不能正常工作。

⑵ DS18B20工作原理

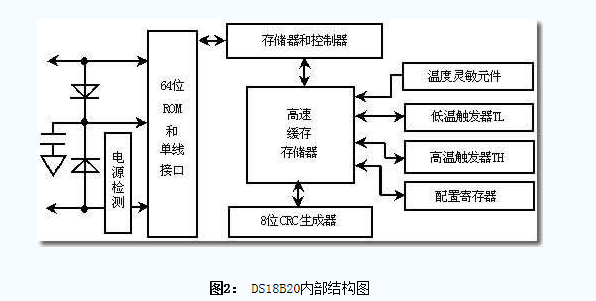


图3-3 DS18B20内部结构图

DS18B20的读写时序和测温原理与DS1820相同，只是得到的温度值的位数因分辨率不同而不同，且温度转换时的延时时间由2s减为750ms。 低温度系数晶振的振荡频率受温度影响很小，用于产生固定频率的脉冲信号送给计数器1。高温度系数晶振随温度变化其振荡率明显改变，所产生的信号作为计数器2的脉冲输入。计数器1和温度寄存器被预置在－55℃所对应的一个基数值。计数器1对低温度系数晶振产生的脉冲信号进行减法计数，当计数器1的预置值减到0时，温度寄存器的值将加1，计数器1的预置将重新被装入，计数器1重新开始对低温度系数晶振产生的脉冲信号进行计数，如此循环直到计数器2计数到0时，停止温度寄存器值的累加，此时温度寄存器中的数值即为所测温度。

⑶ DS18B20供电方式

在外部电源供电方式下，DS18B20工作电源由VDD引脚接入，此时I/O线不需要强上拉，不存在电源电流不足的问题，可以保证转换精度，同时在总线上理论可以挂接任意多个DS18B20传感器，组成多点测温系统。注意：在外部供电的方式下，DS18B20的GND引脚不能悬空，否则不能转换温度，读取的温度总是85℃。

外部电源供电方式是DS18B20最佳的工作方式，工作稳定可靠，抗干扰能力强，而且电路也比较简单，可以开发出稳定可靠的多点温度监控系统，比寄生电源方式只多接一根VCC引线。在外接电源方式下，可以充分发挥DS18B20宽电源电压范围的优点，即使电源电压VCC降到3V时，依然能够保证温度量精度。

(4)硬件连接

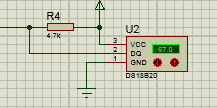


图3-4 DS18B20硬件连接图

**DS18B20引脚定义：**

    (1)DQ为数字信号输入/输出端；   
    (2)GND为电源地；   
    (3)VDD为外接供电电源输入端

**3.1.3显示模块**

LCD1602液晶显示屏

⑴ 简介

1602液晶也叫1602字符型液晶，它是一种专门用来显示字母、数字、符号等的点阵型液晶模块。它由若干个5X7或者5X11等点阵字符位组成，每个点阵字符位都可以显示一个字符，每位之间有一个点距的间隔，每行之间也有间隔，起到了字符间距和行间距的作用，正因为如此所以它不能很好地显示图形（用自定义CGRAM，显示效果也不好）。

1602LCD是指显示的内容为16X2,即可以显示两行共32个字符，每行16个字符液晶模块（显示字符和数字）。微功耗、体积小、显示内容丰富、超薄轻巧，常用在袖珍式仪表和低功耗应用系统中。

⑵ 管脚功能

1602采用标准的16脚接口

第1脚：VSS为电源地

第2脚：VCC接5V电源正极

第3脚：V0为液晶显示器对比度调整端，接正电源时对比度最弱，接地电源时对比度最高（对比度过高时会 产生“鬼影”，使用时可以通过一个10K的电位器调整对比度）。

第4脚：RS为寄存器选择，高电平1时选择数据寄存器、低电平0时选择指令寄存器。

第5脚：RW为读写信号线，高电平(1)时进行读操作，低电平(0)时进行写操作。

第6脚：E(或EN)端为使能(enable)端,高电平(1)时读取信息，负跳变时执行指令。

第7～14脚：D0～D7为8位双向数据端。

第15～16脚：空脚或背灯电源。15脚背光正极，16脚背光负极。

⑶读写时序

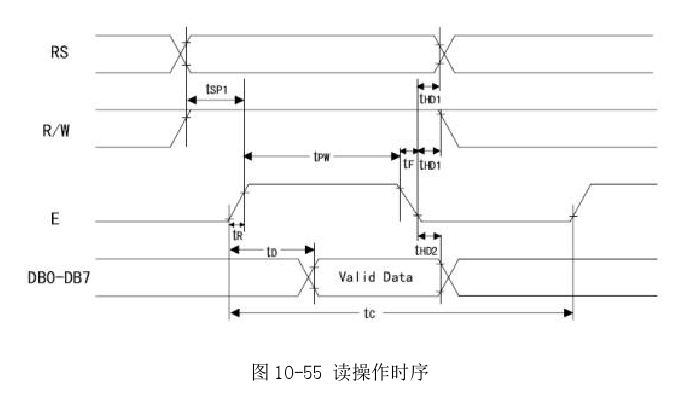


图3-5 读操作时序

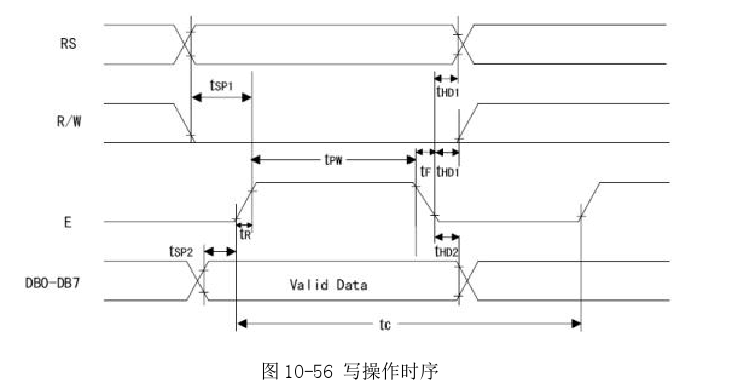
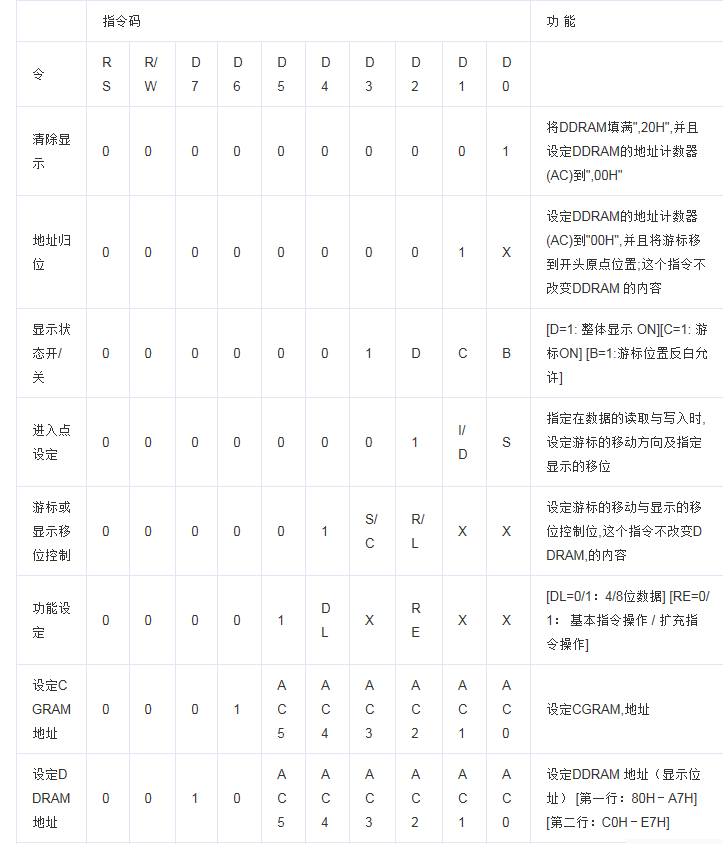
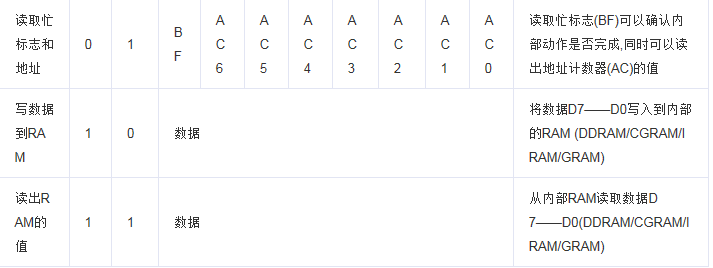


图3-6 写操作时序

⑷ 指令集

表3-1 指令集

⑸硬件连接

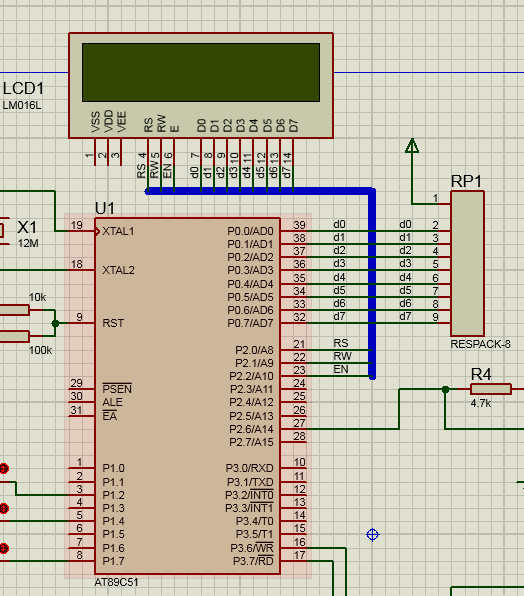


图3-7 LCD1602硬件连接图

**3.1.4 串口通信**

(1)原理

SBUF数据缓冲寄存器这是一个可以直接寻址的串行口专用寄存器。SBUF 包含了两个独立的寄存器，一个是发送寄存，另一个是接收寄存器，但它们都共同使用同一个寻址地址－99H。

SCON串行口控制寄存器通常在芯片或设备中为了监视或控制接口状态，都会引用到接口控制寄存器。SCON 就是51 芯片的串行口控制寄存器。它的寻址地址是98H，是一个可以位寻址的寄存器，作用就是监视和控制51 芯片串行口的工作状态。51 芯片的串口可以工作在几个不同的工作模式下，其工作模式的设置就是使用SCON寄存器。它的各个位的具体定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **SM0 SM1** | **模式** | **功能** | **波特率** |
| 0 0 | 0 | 同步移位寄存器 | fosc/12 |
| 0 1 | 1 | 8位UART | 可变 |
| 1 0 | 2 | 9位UART | fosc/32 或fosc/64 |
| 1 1 | 3 | 9位UART | 可变 |

表3-2 SCON串行口工作模式设置

**SM0、SM1** 为串行口工作模式设置位，这样两位可以对应进行四种模式的设置。串行口工作模式设置。

**SM2** 在模式2、模式3 中为多处理机通信使能位。在模式0 中要求该位为0。

**REM** 为允许接收位，REM 置1 时串口允许接收，置0 时禁止接收。

**TB8** 发送数据位8，在模式2 和3 是要发送的第9位。该位可以用软件根据需要置位或清除，通常这位在通信协议中做奇偶位，在多处理机通信中这一位则用于表示是地址帧还是数据帧。

**RB8** 接收数据位8，在模式2 和3 是已接收数据的第9 位。该位可能是奇偶位，地址/数据标识位。在模式0 中，RB8 为保留位没有被使用。在模式1 中，当SM2=0，RB8 是已接收数据的停止位。

**TI** 发送中断标识位。TI 不会由硬件清除，所以这时必须用软件对其清零。

**RI** 接收中断标识位。

**波特率**：在使用串口做通讯时，一个很重要的参数就是波特率，只有上下位机的波特率一样时才可以进行正常通讯。波特率是指串行端口每秒内可以传输的波特位数。模式1 和模式3 的波特率是可变的，取决于定时器1 或2（52 芯片）的溢出速率。

波特率=（2SMOD÷32）×定时器1 溢出速率

在定时模式1下定时器1 溢出速率=（计数速率）/(256－TH1)

“计数速率”与所使用的晶体振荡器频率有关，在51 芯片中定时器启动后会在每一个机器周期使定时寄存器TH的值增加一，一个机器周期等于十二个振荡周期，所以可以得知51 芯片的计数速率为晶体振荡器频率的1/12，一个12M 的晶振用在51 芯片上，那么51 的计数速率就为1M。

CPU先把准备写入串行设备的数据放到UART的寄存器（临时内存块）中，再通过FIFO（First Input First Output，先入先出队列）传送到串行设备，当串行发送完毕后，将在标志位 TI 置 1，同样，当收到了数据后，也会在 RI 置 1。

UART是一种通用串行数据总线，用于异步通信。该总线双向通信，可以实现全双工传输和接收。

芯片1串口中断 ，采用方式1,8位异步收发，由定时器T1溢出率和SMOD决定波特率（2SMOD/32）\*T1溢出率=4800,根据所测温度和设定值向芯片2发送标记,当前温度大于最高温度时，小于最低温度，及温度正常三种情况。

(2)硬件连接

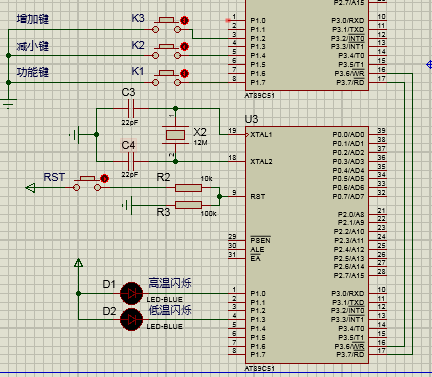


图3-8 串行连接

**3.1.5 报警模块**

(1)功能设计

芯片2接收从芯片1发送的标记，判断当前温度处于三种情况中的哪一种，而控制LED灯和电机的状态。

(2)硬件连接

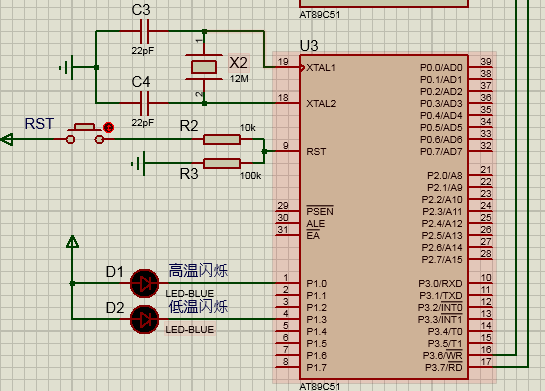
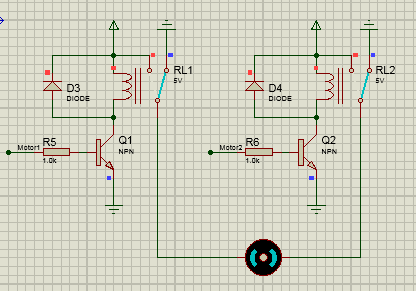


图3-8 LED灯硬件连接

图3-9 电机硬件连接

**3.1.6 按键模块**

(1)功能设计

K1功能键： 按一下开始设置最高温度，再按两下开始设置最低温度，按三下返回显示当前温度，来回反复；

K2增加键： 设置温度增加；

K3减小键： 设置温度减小。

(2)硬件连接

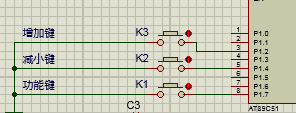


图3-9 按键硬件连接

**3.2 软件设计**

**3.2.1流程图**

1. Project\_LCD

**主函数**  **串口处理**

串口处理

当前温度>上限?

发送’A’

当前温度<下限?

发送’B’

发送’0’

开始

初始化

显示处理

按键处理

串口处理

N

Y

N

Y

**计数器0**

计数器0

1ms?

计数器+1

**按键处理** **显示处理**

Y

Y

N

显示处理

100ms?

当前主页?

获取当前温度并显示

当前设置

下限?

显示下限

显示光标

显示上限

显示光标

当前设置

上限?

Y

Y

N

Y

N

Y

N

按键处理

200ms?

按键0?

菜单切换

按键1?

上限/下限

减1

上限/下限

加1

按键2?

Y

N

Y

N

1. Project\_MOTOR

**主函数**   **温度处理**

开始

初始化

温度处理

LED2状态取反

LED1灭

电机反转

温度处理

200ms?

接受到’A’?

LED1状态取反

LED2灭

电机正转

接收到’B’?

LED2灭

LED1灭

电机停转

显示处理

100ms?

当前主页?

获取当前温度并显示

当前设置

下限?

显示下限

显示光标

显示上限

显示光标

当前设置

上限?

N

Y

N

Y

**计数器0**

计数器0

1ms?

计数器+1

N

N

Y

Y

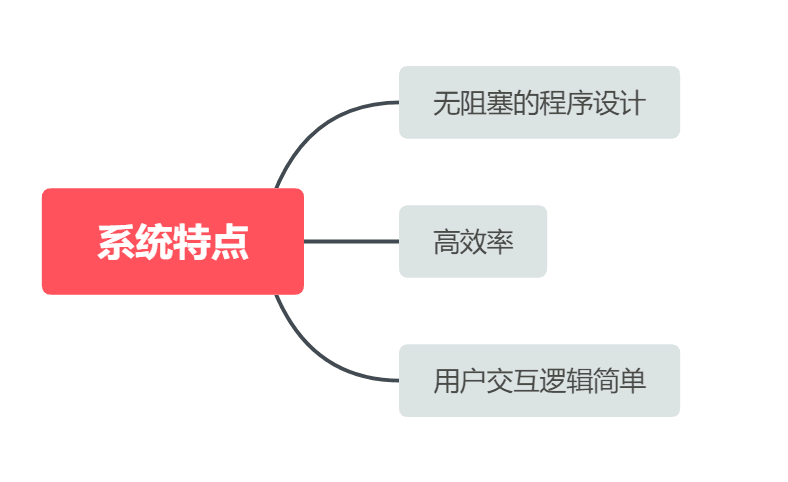
**UART中断**

UART中断

保存接收到的数据

计数器+1

**3.2.2系统特点**

**3.2.3程序设计**

1. **Project\_LCD**

#include <reg51.h>

#include <intrins.h>

#include <stdio.h>

typedef unsigned int uint;

typedef unsigned char uchar;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*引脚定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define LCD\_DATA P0

sbit LCD\_RS = P2^0;

sbit LCD\_RW = P2^1;

sbit LCD\_EN = P2^2;

sbit KEY\_1 = P1^7; //功能键

sbit KEY\_2 = P1^4; //减少键

sbit KEY\_3 = P1^2; //增加按键

sbit DS18B20 = P2^6; //电机使能端

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*全局变量\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define KeyTime 200 //200 \* 1ms = 200ms

#define DispTime 100 //100 \* 1ms = 100ms

volatile uchar KEY\_1ms = 0;

volatile uchar Disp\_1ms = 0;

volatile uchar Menu = 0; //0(显示当前温度), 1(设置最低温度), 2(设置最低温度)

volatile int MaxTemp = 40;

volatile int MinTemp = -10;

volatile float Temperature = 0;

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*函数声明\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_us(uchar x);

void Delay\_ms(uchar x);

void Timer0Init(void);

void KEY\_Proc(void);

void DispTemp(void);

void LCD\_WriteCommand(uchar com);

void LCD\_WriteData(uchar dat);

void LCD\_Init(void);

void LCD\_Show\_Home(void);

void LCD\_Show\_Setting(void);

void DS18B20\_Init(void);

uchar DS18B20\_ReadByte(void);

void DS18B20\_WriteByte(uchar dat);

float DS18B20\_ReadTmp(void);

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*主函数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main(void)

{

LCD\_Init();

Temperature = DS18B20\_ReadTmp();

Delay\_ms(200);

Delay\_ms(200);

Timer0Init();

LCD\_Show\_Home();

while(1)

{

DispTemp();

KEY\_Proc();

}

}

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*延时函数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_us(uchar x) //@12.000MHz

{

while(x--)

{

\_nop\_();

\_nop\_();

\_nop\_();

\_nop\_();

}

}

void Delay\_ms(uchar x) //@12.000MHz

{

uchar i, j;

while(x--)

{

\_nop\_();

\_nop\_();

i = 12;

j = 169;

do

{

while (--j);

} while (--i);

}

}

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*计数器0模块\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Timer0Init(void)

{

TMOD |= 0X01; //选择为定时器0模式，工作方式1，仅用TR0打开启动。

TH0 = 0xFC; //给定时器赋初值，定时1ms 65536-1ms/1us

TL0 = 0x18;

ET0 = 1; //打开定时器0中断允许

EA = 1; //打开总中断

TR0 = 1; //打开定时器

}

void Timer0(void) interrupt 1

{

TH0 = 0xFC; //给定时器赋初值，定时1ms

TL0 = 0x18;

KEY\_1ms ++;

Disp\_1ms ++;

}

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*按键模块\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void KEY\_Proc(void)

{

if (KEY\_1ms >= KeyTime)

{

KEY\_1ms = 0;

if (KEY\_1 == 0)

{

Menu ++;

LCD\_Show\_Setting();

if (Menu == 3)

{

Menu = 0;

LCD\_Show\_Home();

}

}

else if (KEY\_2 == 0)

{

if (Menu == 1)

{

MinTemp --;

}

else if (Menu == 2)

{

MaxTemp --;

if ((MaxTemp - 1) == MinTemp)

{

MaxTemp ++;

}

}

}

else if (KEY\_3 == 0)

{

if (Menu == 1)

{

MinTemp ++;

if ((MaxTemp - 1) == MinTemp)

{

MinTemp --;

}

}

else if (Menu == 2)

{

MaxTemp ++;

}

}

}

}

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*温度显示函数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void DispTemp(void)

{

if (Disp\_1ms >= DispTime)

{

uchar i;

uchar Disp[6] = " ";

uchar Disp1[4] = " ";

Disp\_1ms = 0;

if (Menu == 0)

{

LCD\_WriteCommand(0xC7);

Temperature = DS18B20\_ReadTmp();

sprintf(Disp, "%.1f", Temperature);

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

{

LCD\_WriteData(Disp[i]);

}

}

else if (Menu == 1)

{

LCD\_WriteCommand(0x80);

LCD\_WriteData('>');

LCD\_WriteCommand(0xC0);

LCD\_WriteData(' ');

LCD\_WriteCommand(0x8B);

sprintf(Disp, "%d", MinTemp);

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

{

LCD\_WriteData(Disp[i]);

}

}

else if (Menu == 2)

{

LCD\_WriteCommand(0x80);

LCD\_WriteData(' ');

LCD\_WriteCommand(0xC0);

LCD\_WriteData('>');

LCD\_WriteCommand(0xCB);

sprintf(Disp, "%d", MaxTemp);

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

{

LCD\_WriteData(Disp[i]);

}

}

}

}

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LCD模块\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD\_WriteCommand(uchar com)

{

LCD\_EN = 0; //使能

LCD\_RS = 0; //选择发送命令

LCD\_RW = 0; //选择写入

LCD\_DATA = com; //放入命令

Delay\_us(1); //等待数据稳定

LCD\_EN = 1; //写入时序

Delay\_us(5); //保持时间

LCD\_EN = 0;

}

void LCD\_WriteData(uchar dat)

{

LCD\_EN = 0; //使能清零

LCD\_RS = 1; //选择输入数据

LCD\_RW = 0; //选择写入

LCD\_DATA = dat; //写入数据

Delay\_us(1);

LCD\_EN = 1; //写入时序

Delay\_us(5); //保持时间

LCD\_EN = 0;

}

void LCD\_Init(void)

{

LCD\_WriteCommand(0x38); //开显示

LCD\_WriteCommand(0x0C); //开显示不显示光标

LCD\_WriteCommand(0x06); //写一个指针加1

LCD\_WriteCommand(0x01); //清屏

LCD\_WriteCommand(0x80); //设置数据指针起点

}

void LCD\_Show\_Home(void)

{

uchar i;

uchar Disp[16];

LCD\_WriteCommand(0x80);

sprintf(Disp, " Current Temp: ");

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

{

LCD\_WriteData(Disp[i]);

}

LCD\_WriteCommand(0xC0);

sprintf(Disp, " Temp: C ");

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

{

LCD\_WriteData(Disp[i]);

}

LCD\_WriteCommand(0xCD);

LCD\_WriteData(0xDF);

}

void LCD\_Show\_Setting(void)

{

uchar i;

uchar Disp[16];

LCD\_WriteCommand(0x80);

sprintf(Disp, " Min Temp: %d ", MinTemp);

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

{

LCD\_WriteData(Disp[i]);

}

LCD\_WriteCommand(0xC0);

sprintf(Disp, " Max Temp: %d ", MaxTemp);

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

{

LCD\_WriteData(Disp[i]);

}

}

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DS18B20模块\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void DS18B20\_Init(void)

{

DS18B20 = 1;

Delay\_us(1);

DS18B20 = 0;

Delay\_us(40);

DS18B20 = 1;

Delay\_us(11);

}

uchar DS18B20\_ReadByte(void)

{

uchar i,dat=0;

DS18B20 = 1;

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

{

DS18B20 = 1;

Delay\_us(1);

DS18B20 = 0;

dat >>= 1;

DS18B20 = 1;

if(DS18B20)

dat |= 0X80;

Delay\_us(2);

}

return dat;

}

void DS18B20\_WriteByte(uchar dat)

{

uchar i;

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

{

DS18B20 = 0;

DS18B20 = dat& 0x01;

Delay\_us(2);

DS18B20 = 1;

dat >>= 1;

}

Delay\_us(2);

}

float DS18B20\_ReadTmp(void)

{

float Temp;

uint Temp\_Value[]={0, 0};

DS18B20\_Init();

DS18B20\_WriteByte(0xCC);

DS18B20\_WriteByte(0x44);

Delay\_us(20);

DS18B20\_Init();

DS18B20\_WriteByte(0xCC);

DS18B20\_WriteByte(0xBE);

Temp\_Value[0] = DS18B20\_ReadByte();

Temp\_Value[1] = DS18B20\_ReadByte();

if((Temp\_Value[1]&0xF8)==0xF8)

{

Temp = -1 \* (128 - 0.0625 \* (Temp\_Value[0] | ((Temp\_Value[1]&0x07)<<8)));

}

else

{

Temp = 0.0625 \* (Temp\_Value[0] | ((Temp\_Value[1]&0x07)<<8));

}

return Temp;

}

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

1. **Project\_MOTOR**

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

\* 串口通信实验 \*

实现现象：下载程序后打开串口调试助手，将波特率设置为4800，选择发送的数据就可以显示

在串口助手上。

注意事项：无。

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

#include "reg52.h" //此文件中定义了单片机的一些特殊功能寄存器

typedef unsigned int u16; //对数据类型进行声明定义

typedef unsigned char u8;

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

\* 函数名 :UsartInit()

\* 函数功能 :设置串口

\* 输入 : 无

\* 输出 : 无

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

void UsartInit()

{

SCON=0X50; //设置为工作方式1

TMOD=0X20; //设置计数器工作方式2

PCON=0X80; //波特率加倍

TH1=0XF3; //计数器初始值设置，注意波特率是4800的

TL1=0XF3;

ES=1; //打开接收中断

EA=1; //打开总中断

TR1=1; //打开计数器

}

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

\* 函 数 名 : main

\* 函数功能 : 主函数

\* 输 入 : 无

\* 输 出 : 无

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

void main()

{

UsartInit(); // 串口初始化

while(1);

}

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

\* 函数名 : Usart() interrupt 4

\* 函数功能 : 串口通信中断函数

\* 输入 : 无

\* 输出 : 无

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

void Usart() interrupt 4

{

u8 receiveData;

receiveData=SBUF; //出去接收到的数据

RI = 0 ; //清除接收中断标志位

SBUF=receiveData; //将接收到的数据放入到发送寄存器

while(!TI); //等待发送数据完成

TI=0; //清除发送完成标志位

}

结论

参考文献

1. 刘训飞，陈希，张宇峰，王栋. 单片机技术及应用. 清华大学出版. 2014,6
2. ATC89C51中文使用手册
3. DS18B20数据手册-中文版
4. LCD