****

课程设计报告书

**题目：基于单片机的温度监控系统**

**学 院 电子与信息学院**

**专 业 信息工程**

**学生姓名 苏道平 田喆 温传志**

**学生学号 202330363911**

**202230301074**

**202330372051**

**指导教师 傅娟**

**课程编号 035101112**

**课程学分 1.0**

**起始日期 2025/5/22**

|  |  |
| --- | --- |
| 教  师  评  语 | 教师签名：  日期： |
| 成  绩  评  定 |  |
| 备  注 |  |

**基于单片机的温度监控系统**

**一、设计内容**

1. ​**开机：​**​ LCD屏幕显示“Set\_Alarm:”，等待输入。
2. ​**设定阈值：​**​ 用数字键盘输入报警温度阈值（S1~S10键输入）。
3. ​**切换模式：​**​ 按S11键在“自动”和“手动”模式间切换（LCD显示）。
   * ​**自动模式（Auto Mode）：不能手动解除声光警报​**​。
   * ​**手动模式（Manual Mode）：能手动解除声光警报​**​。
4. ​**开始监控：​**​ 按确认键（S16）进入主循环。
5. ​**显示温度：​**​ LCD同时显示设定的报警值和实时检测到的温度值（带℃单位）。
6. ​**报警判断：​**​ 持续检测温度。
   * ​**超温：​**​蜂鸣器响，LED闪烁，风扇工作。
   * ​**未超温：​**​ 继续循环监控。
7. ​**报警恢复：​**​ 温度回落到阈值以下后，停止声光报警，风扇关闭。

**二、系统总体设计**

2.1、硬件设计原理图

2.1.1.开发板：普中科技51单片机开发板

2.1.2.微控制器

型号：STC89C52RC

频率：11.0592MHz

2.1.3.显示器模块

型号：LCD1602

颜色：蓝色

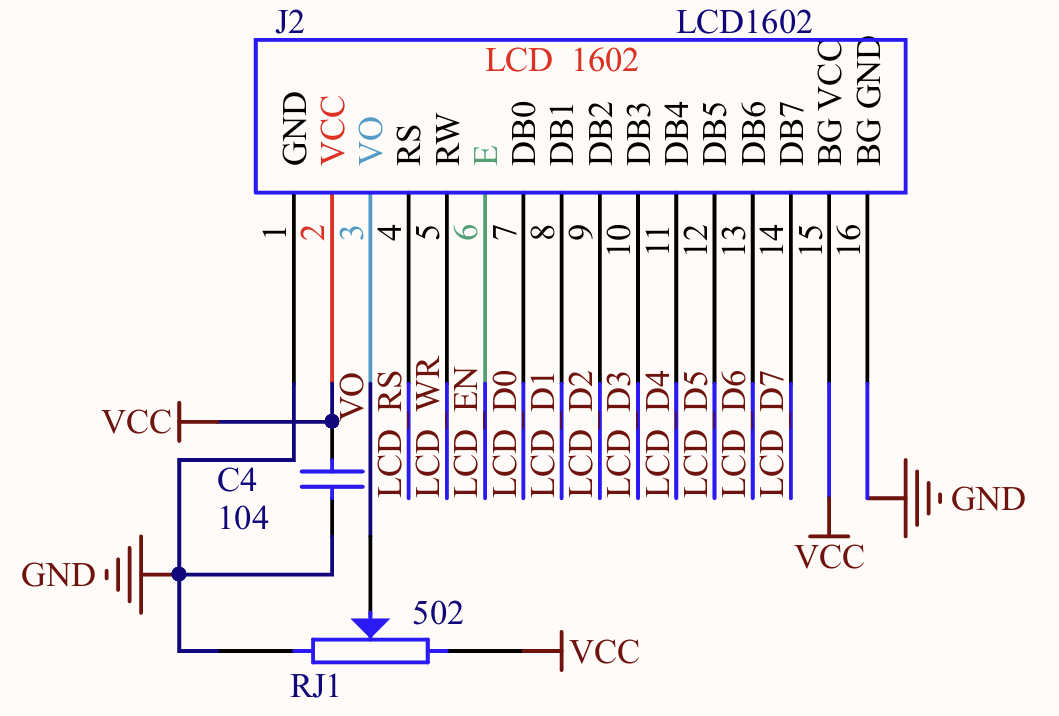


图2.1.1 LCD1602

利用代码来让显示屏显示系统模式、设定温度、阈值温度。

2.1.4.温度传感器模块

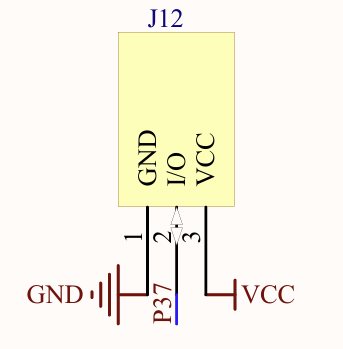
型号：DS18B20

图2.1.2 DS18B20

内部包含一个温度传感器、一个64位的序列号以及一个EEPROM，能够以数字信号的形式从I/O口输出当前温度。

2.1.5.矩阵按键模块

十六个按键：S1、S2、S3、S4、S5、S6、S7、S8、S9、S10、S11、S12、S13、S14、S15、S16

按键类型：4×4矩形按键

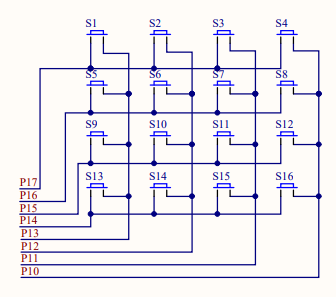


图2.1.3 矩阵按键模块

按下S0-S10获取对应ASCII码，S11切换模式，S16进入温度检测模式，当在手动模式下，摁下任意按键可主动退出温度报警模式。

2.1.6.蜂鸣器模块

驱动芯片：ULN2003D

蜂鸣器：无源蜂鸣器

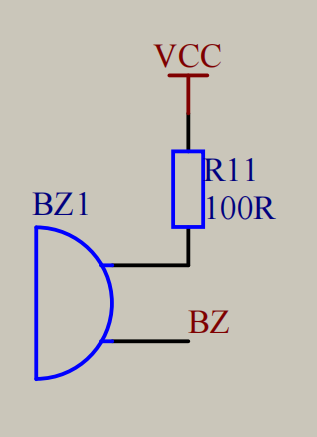
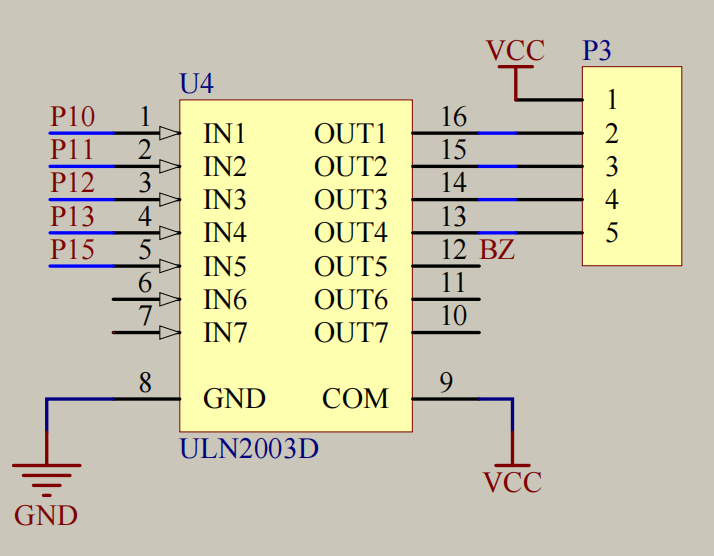


图2.1.4 蜂鸣器和ULN2003D模块

无源蜂鸣器需要一定频率的脉冲信号才能够发生。这里使用单片机的P1.5管脚进行信号脉冲的控制，利用芯片ULN2003D来对控制信号进行放大，从而驱动蜂鸣器的发声。

2.1.7 风扇模块

需要模块：DAC模块和DC5V-3.3V模块

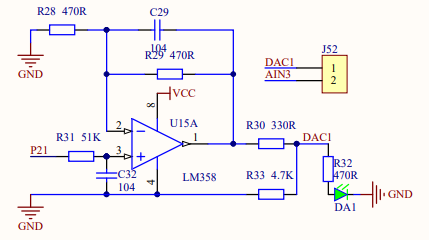


图2.1.5 DAC模块

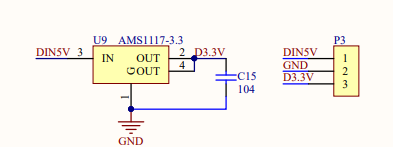


图2.1.6 降压模块

利用降压电源将5V电源转成3.3V给风扇供电，使其工作更稳定，并通过PWM波的有无控制风扇的开关。

2.2、总体软件设计流程图

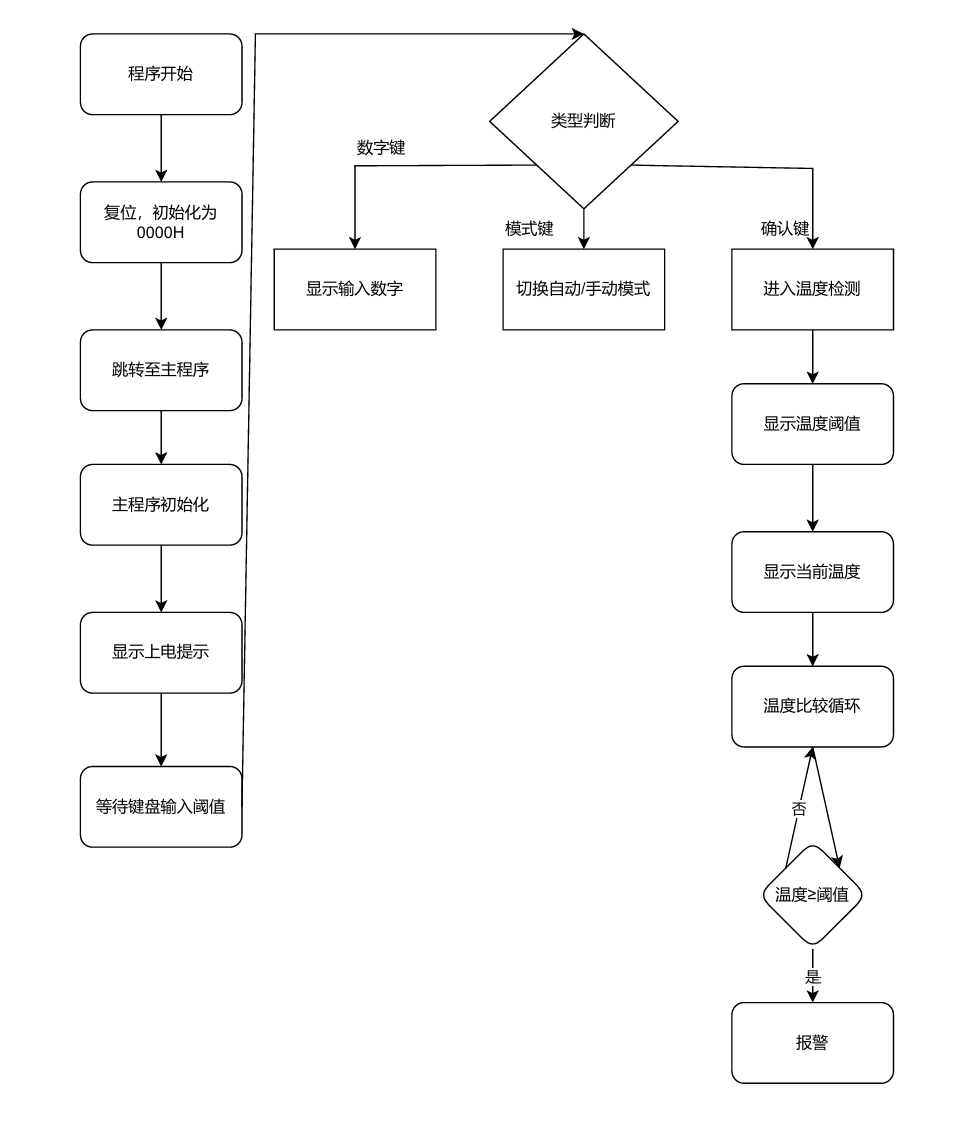
**2.2.1 总体流程图**

图2.2.1 总体流程图

**2.2.2 模块设计流程图**

**2.2.2.1 MAIN程序**

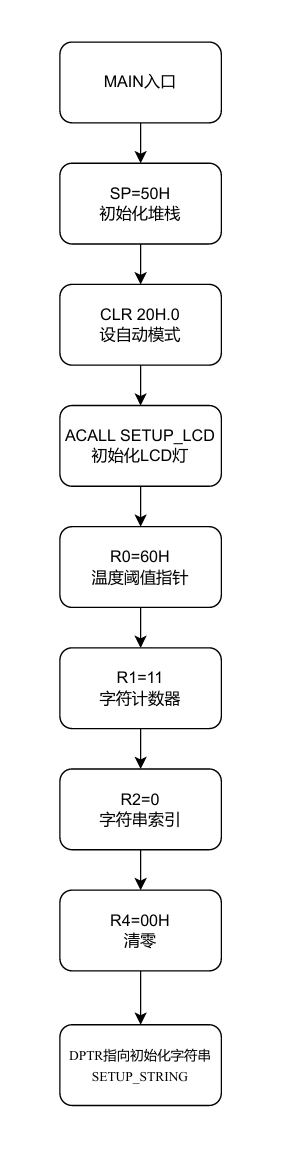


图2.2.1.1 MAIN程序流程图

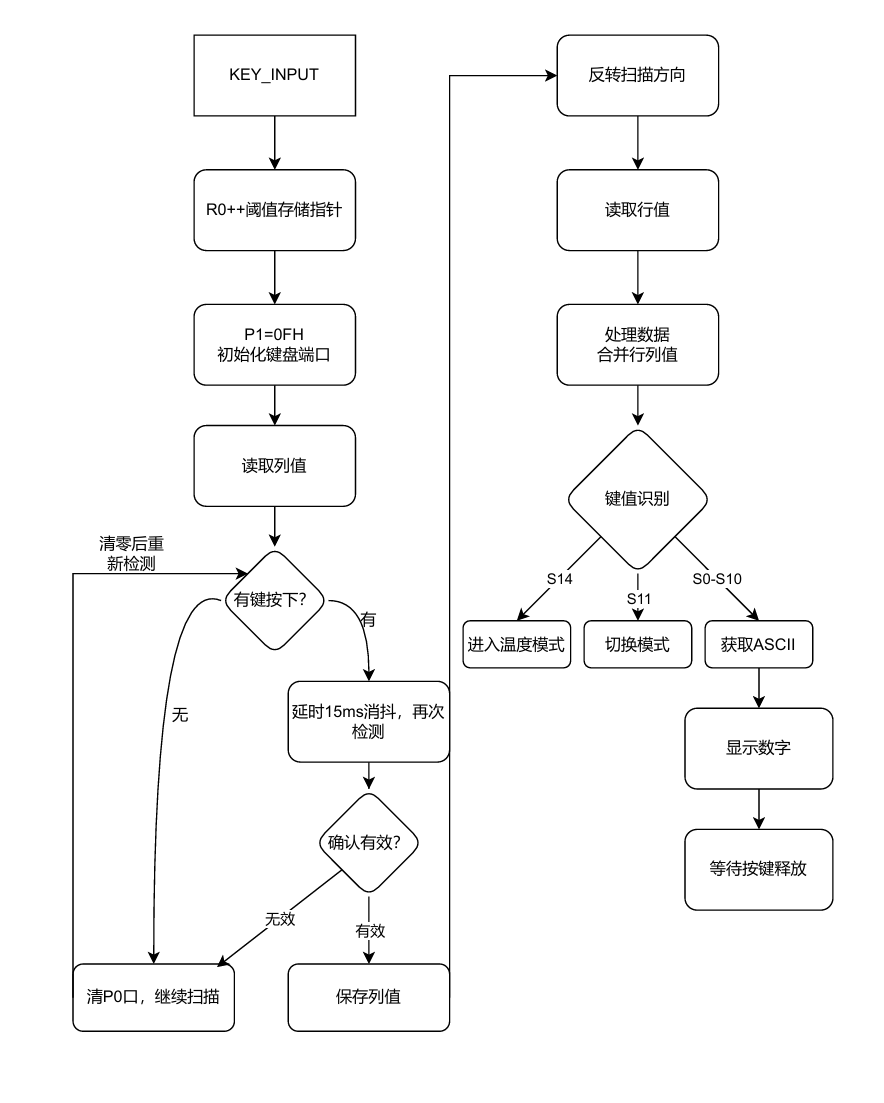
**2.2.2.2 键盘扫描流程**

图2.2.2 键盘扫描程序流程图

**2.2.2.3 温度监测主循环**

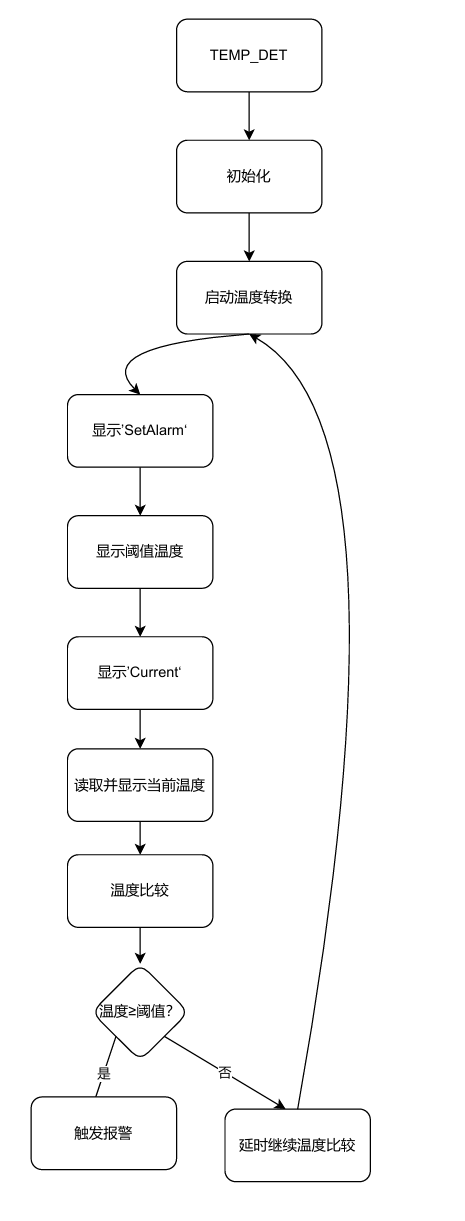


图2.2.2.3温度监测主循环流程图

**2.2.2.4 报警处理模块**

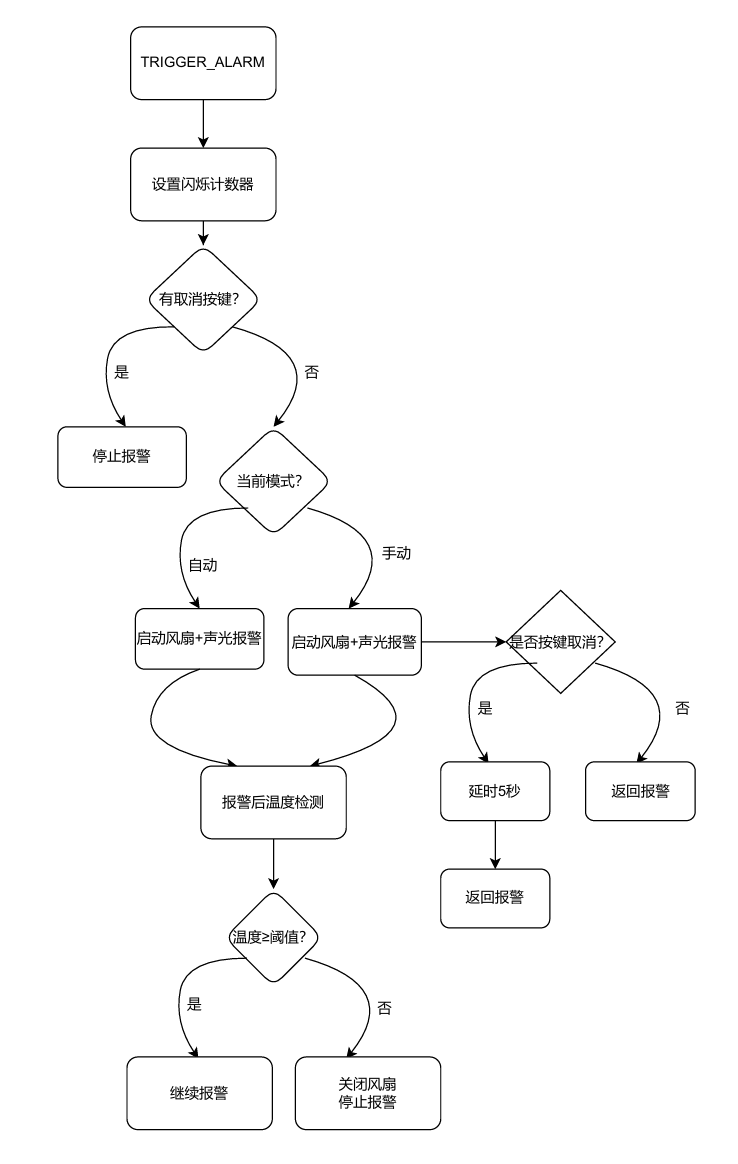
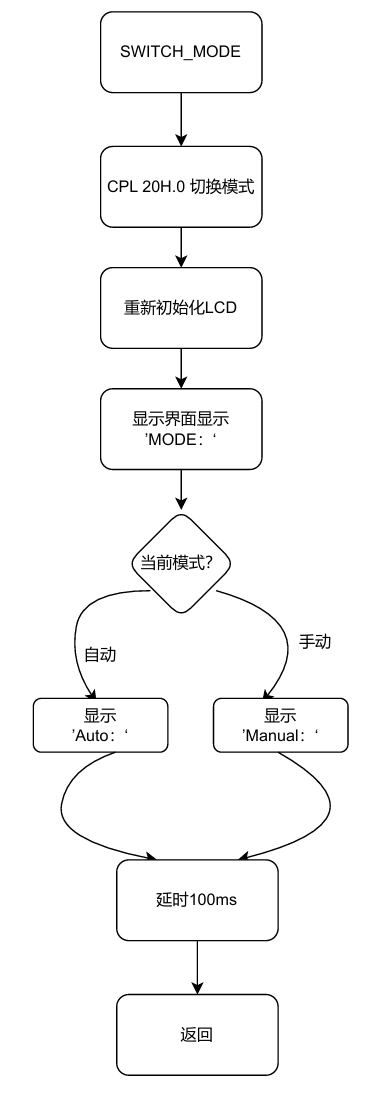


图2.2.2.4报警处理流程图

**2.2.2.5 模式切换模块**

****

**图2.2.2.5模式切换流程图**

**三、分立模块设计**

1. **MAIN程序**

首先初始化8051的堆栈，以#50H为栈底，位地址 20H.0 用作“模式标志”，0为自动模式，1为手动模式。然后初始化LCD屏幕，将字符串“Set\_Alarm:”输出到屏幕上显示，功能是完成程序的初始化，进入到温度阈值设置模式，让用户输入想要输入的值。

1. **键盘扫描程序**

本程序分为5个过程：初始化与持续扫描、消抖处理、行列特征码获取、按键识别、显示与释放检测。

(1)初始化与持续扫描阶段。将P1口高4位设为输出（输出0），低4位设为输入（带上拉）。此时高4位输出低电平，低4位读取列状态。再读取P1口低4位，若值不等于0FH（即某列被拉低），说明有按键按下，跳转至消抖段；否则循环扫描。

​ (2)消抖处理阶段。调用延时函数来延时15ms，避开按键抖动期，然后再次读取P1口低4位，若仍非0FH，确认按键有效，进入行列特征码获取阶段；否则视为抖动，返回扫描循环。

​(3)行列特征码获取阶段。保存列值​通过将低4位列值暂存至R3来实现。接着反转扫描方向​，将高4位设为输入，低4位设为输出（输出0）。读取高4位行值，与列值合并为8位特征码，存入B寄存器。

表1 S1~S16按键的特征码表

| **​按键​** | **​位置（行,列）​​** | **​特征码​** | **​对应值​** |
| --- | --- | --- | --- |
| ​**S1**​ | 行1, 列1 | 77H | 数字 1 |
| ​**S2**​ | 行1, 列2 | 7BH | 数字 2 |
| ​**S3**​ | 行1, 列3 | 7DH | 数字 3 |
| ​**S4**​ | 行1, 列4 | 7EH | 数字 4 |
| ​**S5**​ | 行2, 列1 | B7H | 数字 5 |
| ​**S6**​ | 行2, 列2 | BBH | 数字 6 |
| ​**S7**​ | 行2, 列3 | BDH | 数字 7 |
| ​**S8**​ | 行2, 列4 | BEH | 数字 8 |
| ​**S9**​ | 行3, 列1 | D7H | 数字 9 |
| ​**S10**​ | 行3, 列2 | DBH | 数字 0 |
| ​**S11**​ | 行3, 列3 | DDH | 功能键 \* |
| ​**S12**​ | 行3, 列4 | DEH | 功能键 # |
| ​**S13**​ | 行4, 列1 | E7H | 功能键 A |
| ​**S14**​ | 行4, 列2 | EBH | 功能键 B |
| ​**S15**​ | 行4, 列3 | EDH | 功能键 C |
| ​**S16**​ | 行4, 列4 | EEH | 功能键 D |

(4)按键识别阶段。​​通过CJNE指令将特征码与预设值比对，跳转到对应键处理分支来实现特征码的匹配，匹配后加载ASCII码，跳转至DISP\_SET。设置功能键S11​用来切换自动/手动模式，返回扫描循环。功能键S16​用来当作确认键，初始化LCD并跳转温度检测。

​(5)显示与释放检测阶段。将ASCII码压栈保护，调用SHOW\_KEY\_NUM函数在LCD显示数字。 然后重组行列特征码，与原始特征码（B寄存器）比对。若相同说明未释放，循环检测；若不同则已释放，返回扫描循环。释放检测中增加DELAY\_50MS消抖，确保稳定性。

1. **温度监测主循环**

本模块包含以下4个过程：初始化与提示显示，保存并回显阈值，实时温度监控循环​，以及实时检测与报警逻辑。

(1)初始化与提示显示。首先用ACALL SETUP\_TEMP 函数初始化DS18B20（需遵循单总线协议时序，包括复位、ROM命令等）。再通过ACALL CONVERT\_TO\_DEC 发送转换命令，DS18B20将模拟温度转为数字量（典型耗时750ms），启动温度转换。最后通过查表（MOVC A,@A+DPTR）逐字符输出"Set\_Alarm:"至LCD来实现显示阈值提示​。

​ (2)保存并回显阈值​。首先从堆栈弹出键盘输入的高位（十位）和低位（个位）ASCII码，存入外部RAM地址30H-31H，来实现存储阈值​。再依次读取30H-31H中的ASCII码，调用SHOW\_KEY\_NUM在LCD显示，并追加"°C"符号，实现阈值回显​。

​ (3)实时温度监控循环​。先显示当前温度提示​，输出"Current:"至LCD第二行。然后再读取并显示温度​，ACALL GET\_TEMP 从DS18B20获取温度值（返回十位在A、个位在B寄存器），分两次显示并追加"°C"。接着将光标定位优化​，将光标固定在第二行第8列（MOV\_CURSOR），后续仅刷新数值区域，避免全屏闪烁。

​ (4)实时检测与报警逻辑。用CALL\_TRIGGER\_ALARM调用报警子程序（如置位P1.0启动风扇），并持续检测温度​：若仍超限，循环执行报警流程；若恢复正常，退出报警，返回主循环。

1. **报警处理模块**

本模块包含以下4个部分：报警启动阶段，声光报警循环，模式判断与分支处理，以及手动模式下的按键确认。

(1)报警启动阶段。首先是初始化参数​，MOV R5, #12，设置声光报警循环次数（12次 ≈ 1.2秒）。CLR P2.7：初始化控制信号。然后强制启动风扇​，SETB P1.0：无论模式如何，报警时立即开启风扇（用于散热或通风）。

​ (2)声光报警循环。程序代码如下：

ALARM\_LOOP:

SETB P1.0 ; 启动风扇

CLR P2.0 ; LED亮

CLR P2.5 ; 蜂鸣器响

ACALL DELAY\_50MS ; 持续50ms

SETB P2.0 ; LED灭

SETB P2.5 ; 蜂鸣器停

ACALL DELAY\_50MS ; 间隔50ms

DJNZ R5, ALARM\_LOOP ; 循环12次

​ 执行逻辑​是，每次循环产生 ​100ms 的声光脉冲​（50ms亮/响 + 50ms灭/停），12次循环总时长约 ​1.2秒。风扇在整个报警期间持续开启（SETB P1.0未被修改）。

​ (3)模式判断与分支处理​。包括模式检测​，JB 20H.0, MANUAL\_ALARM\_HANDLE：检查位地址20H.0（手动模式标志位）。

​ 然后是自动模式处理​，若标志位为0（自动模式），直接关闭风扇（CLR P1.0）并返回（RET）。接着是手动模式处理​，若标志位为1（手动模式），进入MANUAL\_ALARM\_HANDLE，等待用户按键确认。

​ (4)手动模式下的按键确认。关键代码如下

MANUAL\_ALARM\_HANDLE:

ACALL CHECK\_KEY\_PRESS ; 检测按键

JC ALARM\_CLEARED ; 有按键则跳转至解除

MOV R5, #12 ; 无按键则重启报警循环

SJMP ALARM\_LOOP

​ 按键检测逻辑是​，CHECK\_KEY\_PRESS子程序通过两次消抖确认按键：首次检测到低电平（ANL A, #0FH ≠ 0FH）。延时15ms后再次检测，仍有效则置位C标志（SETB C）。

​ 若解除报警​，则通过ALARM\_CLEARED中调用5次DELAY\_1S（共延时5秒），随后关闭风扇（CLR P1.0）并返回。

1. **模式切换模块**

在这个模块中，包含4个部分：模式标志位切换，LCD界面刷新​，动态模式名称显示​，以及延时确认。

(1)模式标志位切换​

关键代码：SWITCH\_MODE:

CPL 20H.0 ; 翻转位地址20H.0（0→1或1→0）

通过使用8051内部RAM的位寻址区​（20H-2FH），直接操作单比特标志位，节省内存空间。

​ 其中标志含义如下​：

20H.0 = 0：自动模式（默认），20H.0 = 1：手动模式

​ (2)LCD界面刷新​

关键代码：

ACALL SETUP\_LCD ; 清屏并重置光标并显示"Mode: "

MOV DPTR, #MODE\_STRING

MOV R1, #6 ; 字符数

MOV R2, #0 ; 索引初始化

MODE\_DISP:

MOV A, R2

MOVC A, @A+DPTR ; 查表取字符

ACALL SHOW\_STR ; 显示字符

INC R2

DJNZ R1, MODE\_DISP ; 循环显示

​ 然后是字符串处理​，通过查表指令MOVC从程序存储器读取字符（如4DH→'M'），避免硬编码。

​ (3)动态模式名称显示​

关键代码：

JB 20H.0, SHOW\_MANUAL ; 检测模式标志

; 自动模式分支

MOV DPTR, #AUTO\_STRING ; "Auto "

SJMP SHOW\_MODE

SHOW\_MANUAL:

MOV DPTR, #MANUAL\_STRING ; "Manual"

SHOW\_MODE:

MOV R1, #7 ; 字符串长度（含填充空格）

... ; 循环显示剩余字符

​设计​特点是对齐设计，"Auto"后补两个空格（20H），使"Auto"与"Manual"显示宽度一致（7字符），避免残留字符。

​ (4)延时确认​

关键代码：ACALL DELAY\_50MS

ACALL DELAY\_50MS ; 合计100ms延时

RET

​ 此处在交互上做了优化​，延时确保用户看清模式提示，防止快速切换导致视觉混淆。

**四、程序调试与结果分析**

**1. LCD初始化时序问题​**

* **问题**​：LCD1602液晶上电后初始化失败（显示乱码/不响应）
  + 原因：初始化指令间隔时间不足，未满足1602的建立时间要求（典型需15ms以上）
  + ​**解决方案**​：
    - 在每条关键指令（如清屏、模式设置）后插入 ACALL DELAY\_15MS或 ACALL DELAY\_50MS
    - 严格按手册顺序初始化：清屏→输入模式→显示开关→功能设置（见 SETUP\_LCD子程序）

**​2. 键盘扫描误触发​**

* ​**问题**​：按键抖动导致多次触发或检测到无效键值
  + 原因：无硬件消抖，扫描逻辑未处理抖动和释放
  + ​**解决方案**​：
    - 加入两级消抖：首次检测到按下后延时15ms再确认（k1到 k2段）
    - 按键释放检测：在 DISP\_SET中循环检查特征码变化，确保物理释放后才接受输入

**​3. DS18B20通信失败​**

* ​**问题**​：温度读取值异常
  + 原因：时序偏差（如延时精度不足）、未响应存在脉冲、命令发送顺序错误
  + ​**解决方案**​：
    - 精确延时：用 DELAY\_60US和循环计数确保复位脉冲（480μs）、读写时隙（60μs）的精度
    - 存在脉冲检测：在 SETUP\_TEMP中通过 JB P3.7,CHECK\_EXIST循环等待从机响应
    - 严格按协议顺序：初始化→跳过ROM（CCH）→启动转换（44H）→读暂存器（BEH）

**​4. 温度阈值比较逻辑错误​**

* ​**问题**​：报警条件判断失效（如未触发或误触发）
  + 原因：未将ASCII码阈值转为二进制数值，直接比较ASCII码
  + ​**解决方案**​：
    - 在 CMP\_TEMP子程序中调用 CONVERT\_ASCI\_TO\_DEC
    - 将键盘输入的ASCII阈值（如"35"）转为二进制数值（0x23）再与实测温度比较

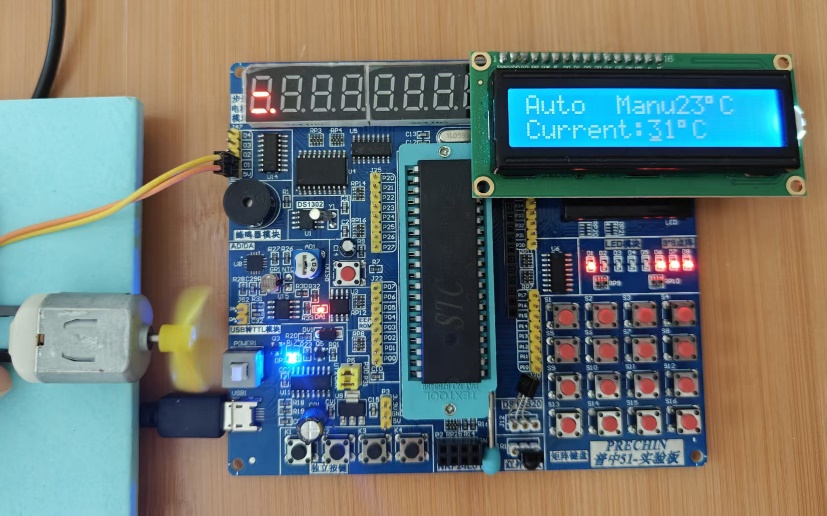
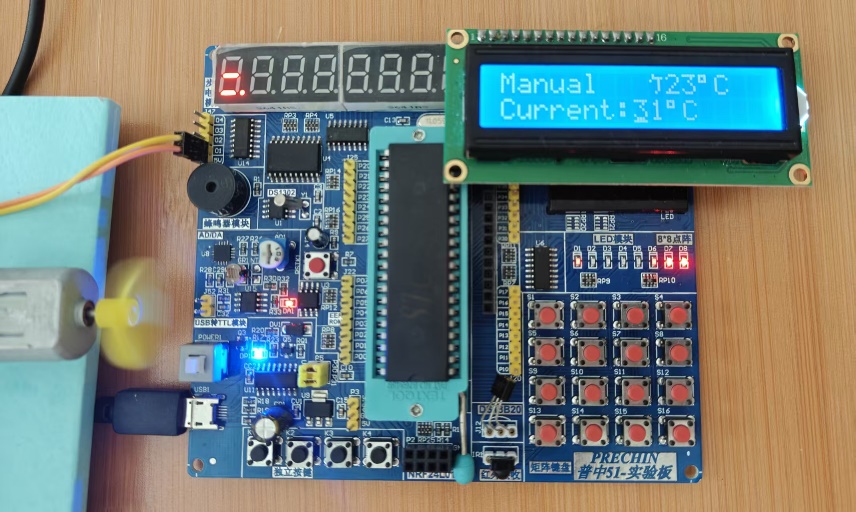
产品各功能的演示效果：

图4.1 自动模式下报警

自动模式报警：当前温度31℃高于手动设定的阈值温度23℃，触发声光报警，风扇工作，按4×4键盘上的任意键不能解除声光报警。只有当环境温度降到阈值以下时，才会停止报警，风扇停止工作。

图4.2 手动模式下报警

手动模式报警：当前温度31℃高于手动设定的阈值温度23℃，触发声光报警，风扇工作，按4×4键盘上的任意键可以解除声光报警，解除报警后风扇继续工作。5s后若环境温度仍高于阈值温度，再次触发声光报警。

**五、课程设计总结**

自评成绩：优秀

自评成绩理由：

1. **功能完整性强：​**​
   * 实现了从键盘输入、LCD显示、温度采集（DS18B20）、阈值比较到声光报警（LED+蜂鸣器）和风扇控制（P1.0）的完整闭环。
   * 支持自动/手动两种模式切换（通过S11键）。
   * 自动模式：超温报警后，温度降至阈值以下自动解除报警和风扇。
   * 手动模式：超温报警后，需用户按键确认才能解除报警（有5秒延时）。
2. ​**模块化清晰：​**​
   * 主程序流程清晰：初始化 -> 设置阈值 -> 温度检测循环 -> 是否启动报警。
   * 关键功能封装成子程序：DELAY\_\*(多种延时), SETUP\_LCD, SHOW\_STR, SHOW\_KEY\_NUM, NEXTLINE, MOV\_CURSOR, SETUP\_TEMP, WRITE\_DS18B20, GET\_TEMP, CMP\_TEMP, TRIGGER\_ALARM, CHECK\_KEY\_PRESS, CONVERT\_ASCII\_TO\_DEC等。代码复用性高。
3. ​**资源规划合理：​**​
   * 堆栈指针初始化 (MOV SP, #50H)。
   * 使用位地址20H.0作为清晰的状态标志位（自动/手动模式）。
   * 合理规划外部RAM地址存放阈值 (30H, 31H- ASCII码)和温度原始值 (70H开始)及中间结果 (7AH)。
   * I/O口分配明确：P0-LCD数据，P2-LCD控制+LED/蜂鸣器，P1-键盘+风扇，P3.7-DS18B20单总线。

心得与总结：

在基于普中51单片机的温度自动报警器开发中，我们团队分工明确且紧密协作：​苏道平负责搭建系统主框架（MAIN程序），实现键盘扫描功能，确保按键输入准确响应；田喆承担键盘检测主循环的逻辑设计，并完成报警处理模块开发，实现超限温度的声光报警及LCD报警状态提示；温传志则负责模式切换状态机（如正常监控、阈值设置模式）和延时子程序模块，为传感器时序、按键消抖提供精准时间控制。三人模块通过接口联动（如温传志的延时函数支撑苏道平的键盘消抖，田喆的报警逻辑依赖温传志的模式参数），最终实现温度监测的完整系统。

经过了这次的课设，我们对于单片机的理解深刻了许多。之前用C语言的时候，许多初始化的代码都是一知半解，更多的都是注重在逻辑的把控上。这次运用我们所学的汇编，不仅逻辑要把握住，单片机内部的初始化寄存器应该如何去设置，去选择模式或者工作方式，或者都要比较清楚，从底层上知道了单片机如何去工作的，对于以后的单片机使用有很大的帮助。

这次的课设当中也感受到编程是一件很有意思的工作。代码一点点的被编写出来，被调试出来，最终实现了我们想要实现的功能，这是一件非常有成就感的事情。它是一种把我们所学去实践去应用的过程，在同时还有现象的反馈，从而激励我们进行下一步的调整，如何去找出bug。

**六、程序清单附录**

; 程序起始地址设定，ORG伪指令指定程序从0000H地址开始执行

ORG 0000H

; 上电后，单片机立即无条件跳转到主程序入口MAIN，跳过中断向量区

AJMP MAIN

;=====================================================

; 主程序真正入口地址，ORG 0200H避开中断向量区，确保主程序不会被中断向量打断

ORG 0200H

;-----------------------------------------------------

; 主程序开始，标签MAIN为程序入口点

MAIN:

; 初始化8051堆栈指针SP，#50H为栈底，防止堆栈溢出影响其他寄存器

MOV SP,#50H

; 位地址20H.0用作"模式标志"：0=自动模式（默认），1=手动模式

; 上电默认为自动模式，因此先清零该位确保初始状态为自动

CLR 20H.0

; 调用LCD初始化子程序SETUP\_LCD，完成1602液晶的基本配置，确保显示正常

ACALL SETUP\_LCD

; 以下为在LCD第一行显示固定提示"Set\_Alarm:"，提示用户输入温度阈值

MOV R0,#60H ; R0暂存显示缓冲区指针，指向缓冲区起始地址60H

MOV R1,#11 ; 字符串长度11字节（含空格）

MOV R2,#0 ; 索引从0开始，用于逐字符显示

MOV R4,#00H ; R4初始化，暂未使用，保留备用

MOV P2,#1FH ; P2口初始化，准备控制LCD的RS/RW/E信号线

MOV DPTR,#SETUP\_STRING ; DPTR指向"Set\_Alarm:"字符串表头，准备查表显示

; 循环显示字符串SETUP\_STRING中的每个字符

SETUP\_DISP:

MOV A,R2 ; 取当前索引

MOVC A,@A+DPTR ; 查表取字符，根据索引从字符串表中读取字符

ACALL SHOW\_STR ; 调用子程序SHOW\_STR，将字符送LCD显示

INC R2 ; 索引+1，准备显示下一个字符

DJNZ R1,SETUP\_DISP ; 若未显示完11个字符则继续循环

; 稍作延时，给人眼辨识时间，确保用户能看清提示信息

ACALL DELAY\_50MS

; 光标定位到LCD坐标(0,10)，即第一行第10列，方便后续输入阈值显示位置

MOV A,#0 ; A=0表示第一行

MOV B,#10 ; B=10表示第10列

ACALL MOV\_CURSOR ; 调用子程序MOV\_CURSOR定位光标

; 跳转至键盘扫描与温度阈值输入阶段，等待用户输入

AJMP KEY\_INPUT

;-----------------------------------------------------

; 固定提示字符串：ASCII码"Set\_Alarm:"，每个字符用十六进制表示

SETUP\_STRING: DB 53H,65H,74H,5FH,41H,6CH,61H,72H,6DH,3AH

;=====================================================

; 4×4键盘扫描与温度阈值输入主循环，持续检测键盘输入

KEY\_INPUT:

INC R0 ; R0自增，计数键盘扫描次数，可用于调试

; 将P1口低4位设为输入模式，高4位输出0，为行列扫描做准备

MOV P1, #0FH

MOV A, P1 ; 读取P1口当前状态

ANL A, #0FH ; 屏蔽高4位，保留低4位列值，检测列线状态

CJNE A, #0FH, k1 ; 若有键按下（低4位不全为1）则跳k1处理，否则继续扫描

; 无键按下时，P0口清0（此处P0可能用于其他指示或调试）

MOV P0,#00H

AJMP KEY\_INPUT ; 无键按下则继续扫描键盘

;-----------------------------------------------------

; 消抖阶段，延时15ms后再次确认按键状态，防止误触发

k1:

ACALL DELAY\_15MS ; 延时15ms，消除机械抖动

MOV P1,#0FH ; 再次设置P1口低4位为输入

MOV A,P1 ; 再次读取P1口状态

ANL A,#0FH ; 再次屏蔽高4位，保留列值

CJNE A,#0FH,k2 ; 若再次确认仍按下（低4位不全为1）则跳k2，否则认为抖动忽略

; 若延时后确认无键按下（抖动），则返回继续扫描

AJMP KEY\_INPUT

;-----------------------------------------------------

; 确认有键按下后，保存行列值以便后续查表确定具体按键

k2:

MOV R3,A ; R3暂存低4位（列值），后续与行值合并

; 反转扫描方向：高4位输出0，低4位输入，检测行线状态

MOV P1,#0F0H ; 高4位输出0，低4位输入

MOV A,P1 ; 读取P1口状态

ANL A,#0F0H ; 取高4位行值，屏蔽低4位

ORL A,R3 ; 行列合并得8位特征码，存入B寄存器

MOV B,A ; 将合并后的特征码存入B，用于后续查表

;=====================================================

; 根据行列特征码查表得到对应按键编号，依次比对S1~S10、S11、S16

; 每个按键对应一个特征码，查表后跳转到相应处理

; 比对S1（特征码77H）

S1:

CJNE A,#77H,S2 ; 若特征码不等于77H则跳S2继续比对

MOV A,#31H ; 若等于77H，得到ASCII码'1'

LJMP DISP\_SET ; 跳到显示/存储该数字的子程序

; 比对S2（特征码7BH）

S2:

CJNE A,#7BH,S3 ; 若特征码不等于7BH则跳S3继续比对

MOV A,#32H ; 若等于7BH，得到ASCII码'2'

LJMP DISP\_SET ; 跳到显示/存储该数字的子程序

; 比对S3（特征码7DH）

S3:

CJNE A,#7DH,S4 ; 若特征码不等于7DH则跳S4继续比对

MOV A,#33H ; 若等于7DH，得到ASCII码'3'

LJMP DISP\_SET ; 跳到显示/存储该数字的子程序

; 比对S4（特征码7EH）

S4:

CJNE A,#7EH,S5 ; 若特征码不等于7EH则跳S5继续比对

MOV A,#34H ; 若等于7EH，得到ASCII码'4'

LJMP DISP\_SET ; 跳到显示/存储该数字的子程序

; 比对S5（特征码B7H）

S5:

CJNE A,#0B7H,S6 ; 若特征码不等于B7H则跳S6继续比对

MOV A,#35H ; 若等于B7H，得到ASCII码'5'

LJMP DISP\_SET ; 跳到显示/存储该数字的子程序

; 比对S6（特征码BBH）

S6:

CJNE A,#0BBH,S7 ; 若特征码不等于BBH则跳S7继续比对

MOV A,#36H ; 若等于BBH，得到ASCII码'6'

LJMP DISP\_SET ; 跳到显示/存储该数字的子程序

; 比对S7（特征码BDH）

S7:

CJNE A,#0BDH,S8 ; 若特征码不等于BDH则跳S8继续比对

MOV A,#37H ; 若等于BDH，得到ASCII码'7'

LJMP DISP\_SET ; 跳到显示/存储该数字的子程序

; 比对S8（特征码BEH）

S8:

CJNE A,#0BEH,S9 ; 若特征码不等于BEH则跳S9继续比对

MOV A,#38H ; 若等于BEH，得到ASCII码'8'

LJMP DISP\_SET ; 跳到显示/存储该数字的子程序

; 比对S9（特征码D7H）

S9:

CJNE A,#0D7H,S10 ; 若特征码不等于D7H则跳S10继续比对

MOV A,#39H ; 若等于D7H，得到ASCII码'9'

LJMP DISP\_SET ; 跳到显示/存储该数字的子程序

; 比对S10（特征码DBH）

S10:

CJNE A,#0DBH,S11 ; 若特征码不等于DBH则跳S11继续比对

MOV A,#30H ; 若等于DBH，得到ASCII码'0'

LJMP DISP\_SET ; 跳到显示/存储该数字的子程序

;-----------------------------------------------------

; S11键：切换自动/手动模式（特征码DDH）

S11:

CJNE A,#0DDH,S16 ; 若特征码不等于DDH则跳S16继续比对

ACALL SWITCH\_MODE ; 调用模式切换子程序SWITCH\_MODE

AJMP KEY\_INPUT ; 切换模式后继续扫描键盘

;-----------------------------------------------------

; S16键：确认键，输入阈值完毕，进入温度检测（特征码EEH）

S16:

CJNE A,#0EEH,JMP\_TO\_KEY\_INPUT ; 若特征码不等于EEH则跳无效键处理

ACALL SETUP\_LCD ; 重新初始化LCD清屏，准备进入温度检测界面

AJMP TEMP\_DET ; 跳转到温度检测主循环TEMP\_DET

; 非S1~S11、S16的其他键均视为无效，继续扫描键盘

JMP\_TO\_KEY\_INPUT:

AJMP KEY\_INPUT

;=====================================================

; 显示刚刚输入的数字，并等待按键释放，防止重复触发

DISP\_SET:

PUSH ACC ; 保护现场，将累加器A（ASCII数字）压栈保存

ACALL SHOW\_KEY\_NUM ; 调用子程序SHOW\_KEY\_NUM，将数字送LCD显示

; 循环检测按键是否释放，确保用户松开按键后才继续

CHECK\_LOOP:

MOV P1,#0F0H ; 再次设置P1口高4位为输出0，低4位输入，检测行列

MOV A,P1 ; 读取P1口状态

ANL A,#0F0H ; 取高4位行值

ORL A,R3 ; 重组8位特征码，与之前存入B的值比较

CJNE A,B,JMP\_TO\_KEY\_INPUT ; 若特征码改变说明按键已释放，跳回继续扫描

; 若特征码未改变，说明按键未释放，延时消抖继续等待

ACALL DELAY\_50MS ; 延时消抖

AJMP CHECK\_LOOP ; 循环等待真正释放

;=====================================================

; 切换自动/手动模式子程序

SWITCH\_MODE:

CPL 20H.0 ; 对模式标志位20H.0取反：0变1或1变0，实现模式切换

ACALL SETUP\_LCD ; 重新清屏LCD，准备显示新模式信息

; 显示"Mode: "字符串，提示用户当前模式

MOV DPTR,#MODE\_STRING ; DPTR指向"Mode: "字符串表头

MOV R1,#6 ; 字符串长度6字节

MOV R2,#0 ; 索引从0开始

; 循环显示"Mode: "字符串

MODE\_DISP:

MOV A,R2 ; 取当前索引

MOVC A,@A+DPTR ; 查表取字符

ACALL SHOW\_STR ; 调用子程序SHOW\_STR显示字符

INC R2 ; 索引+1

DJNZ R1,MODE\_DISP ; 未显示完6字节则继续

ACALL NEXT\_LINE ; LCD换行到第二行，准备显示当前模式

; 根据当前模式显示"Auto"或"Manual"

JB 20H.0,SHOW\_MANUAL ; 若模式标志位为1（手动模式）则跳SHOW\_MANUAL

MOV DPTR,#AUTO\_STRING ; 否则指向"Auto"字符串表头

SJMP SHOW\_MODE ; 跳转到SHOW\_MODE显示自动模式

; 显示手动模式字符串

SHOW\_MANUAL:

MOV DPTR,#MANUAL\_STRING ; 指向"Manual"字符串表头

; 显示当前模式字符串（Auto或Manual）

SHOW\_MODE:

MOV R1,#7 ; "Auto"或"Manual"字符串长度

MOV R2,#0 ; 索引从0开始

; 循环显示当前模式字符串

CURRENT\_MODE\_DISP:

MOV A,R2 ; 取当前索引

MOVC A,@A+DPTR ; 查表取字符

ACALL SHOW\_STR ; 调用子程序SHOW\_STR显示字符

INC R2 ; 索引+1

DJNZ R1,CURRENT\_MODE\_DISP ; 未显示完则继续

; 延时100ms给人眼观察，确保用户看清模式切换结果

ACALL DELAY\_50MS

ACALL DELAY\_50MS

RET ; 子程序返回，继续主程序

;-----------------------------------------------------

; 固定字符串表：ASCII码表示

MODE\_STRING: DB 4DH,6FH,64H,65H,3AH,20H ; "Mode: "

AUTO\_STRING: DB 41H,75H,74H,6FH,20H,20H ; "Auto "

MANUAL\_STRING: DB 4DH,61H,6EH,75H,61H,6CH ; "Manual"

;=====================================================

; 温度检测主循环，持续读取温度并与阈值比较

TEMP\_DET:

ACALL SETUP\_TEMP ; 初始化DS18B20温度传感器，确保通信正常

ACALL CONVERT\_TO\_DEC ; 启动一次温度转换命令，DS18B20开始转换温度

; 显示"Set\_Alarm:"提示（11字节），提醒用户当前设定阈值

MOV R1,#11 ; 字符串长度11字节

MOV R2,#0 ; 索引从0开始

; 循环显示"Set\_Alarm:"

SetAlarm\_DISP:

MOV A,R2 ; 取当前索引

MOVC A,@A+DPTR ; 查表取字符

ACALL SHOW\_STR ; 调用子程序SHOW\_STR显示字符

INC R2 ; 索引+1

DJNZ R1,SetAlarm\_DISP ; 未显示完11字节则继续

ACALL SHIFT\_CURSOR\_LEFT ; 光标左移一位对齐，美观显示

; 将键盘输入的两位阈值ASCII码（先高位后低位）存入外部RAM

MOV R0,#31H ; 指向外部RAM地址31H，准备存储高位（十位）

POP ACC ; 弹出高位（十位）ASCII码

MOVX @R0,A ; 存高位到31H

MOV R0,#30H ; 指向外部RAM地址30H，准备存储低位（个位）

POP ACC ; 弹出低位（个位）ASCII码

MOVX @R0,A ; 存低位到30H

; 回显刚刚输入的温度阈值到LCD，让用户确认输入正确

MOV R0,#30H ; 指向外部RAM地址30H，准备显示低位

MOV R4,#2 ; 共两位（十位和个位）

; 循环显示两位阈值

SET\_DISP\_NUM:

MOVX A,@R0 ; 从外部RAM读取ASCII码

ACALL DELAY\_50MS ; 延时确保显示稳定

ACALL SHOW\_KEY\_NUM ; 调用子程序SHOW\_KEY\_NUM显示一位

ACALL DELAY\_50MS ; 延时确保显示稳定

INC R0 ; 指向下一位

DJNZ R4,SET\_DISP\_NUM ; 未显示完两位则继续

; 显示"°C"符号，表示温度单位

MOV A,#0DFH ; 取"°"符号的ASCII码

ACALL SHOW\_KEY\_NUM ; 显示"°"

MOV A,#43H ; 取"C"符号的ASCII码

ACALL SHOW\_KEY\_NUM ; 显示"C"

ACALL NEXT\_LINE ; LCD换行到第二行，准备显示当前温度

; 显示"Current:"提示（9字节），提示用户当前温度

MOV DPTR,#SHOW\_CURRENT\_TEMP ; DPTR指向"Current:"字符串表头

MOV R1,#9 ; 字符串长度9字节

MOV R2,#0 ; 索引从0开始

; 循环显示"Current:"

CURRENT\_DISP:

MOV A,R2 ; 取当前索引

MOVC A,@A+DPTR ; 查表取字符

ACALL SHOW\_STR ; 调用子程序SHOW\_STR显示字符

INC R2 ; 索引+1

DJNZ R1,CURRENT\_DISP ; 未显示完9字节则继续

ACALL SHIFT\_CURSOR\_LEFT ; 光标左移一位对齐

; 读取一次当前温度并显示

ACALL SETUP\_TEMP ; 初始化DS18B20，准备通信

ACALL GET\_TEMP ; 读取温度并转换为ASCII码

; 显示当前温度十位和个位

CURRENT\_NUM\_DISP:

ACALL DELAY\_50MS ; 延时确保显示稳定

ACALL SHOW\_KEY\_NUM ; 显示十位

MOV A,B ; 将个位ASCII码移入A

ACALL DELAY\_50MS ; 延时确保显示稳定

ACALL SHOW\_KEY\_NUM ; 显示个位

; 显示"°C"符号

MOV A,#0DFH ; 取"°"符号

ACALL SHOW\_KEY\_NUM ; 显示"°"

MOV A,#43H ; 取"C"符号

ACALL SHOW\_KEY\_NUM ; 显示"C"

; 光标定位到(1,8)即第二行第8列，方便实时更新温度显示

MOV A,#1 ; A=1表示第二行

MOV B,#8 ; B=8表示第8列

ACALL MOV\_CURSOR ; 调用子程序MOV\_CURSOR定位光标

;-----------------------------------------------------

; 以下为实时循环：每50ms读取一次温度并比较阈值

GET\_TEMP\_LOOP:

CLR P1.0 ; 关闭风扇（若之前报警已停），确保初始状态

ACALL SETUP\_TEMP ; 初始化DS18B20，准备通信

ACALL CONVERT\_TO\_DEC ; 启动温度转换

ACALL DELAY\_50MS ; 延时等待转换完成

ACALL SETUP\_TEMP ; 再次初始化DS18B20，准备读取

ACALL GET\_TEMP ; 读取温度并转换为ASCII码

ACALL DELAY\_50MS ; 延时确保显示稳定

; 显示当前温度十位和个位，刷新显示

ACALL SHOW\_KEY\_NUM ; 显示十位

MOV A,B ; 将个位ASCII码移入A

ACALL DELAY\_50MS ; 延时确保显示稳定

ACALL SHOW\_KEY\_NUM ; 显示个位

; 光标重新定位到(1,8)，确保更新位置正确

MOV A,#1 ; A=1表示第二行

MOV B,#8 ; B=8表示第8列

ACALL MOV\_CURSOR ; 调用子程序MOV\_CURSOR定位光标

; 比较当前温度与阈值

ACALL CMP\_TEMP ; 调用子程序CMP\_TEMP比较温度

JNC CALL\_TRIGGER\_ALARM ; 若当前温度≥阈值（C=1）则跳报警

SJMP CONTINUE ; 否则继续循环

; 报警处理

CALL\_TRIGGER\_ALARM:

ACALL TRIGGER\_ALARM ; 进入报警子程序TRIGGER\_ALARM

; 报警子程序内会再次比较温度，若仍超限则继续报警

ACALL SETUP\_TEMP ; 再次初始化DS18B20

ACALL CONVERT\_TO\_DEC ; 启动温度转换

ACALL DELAY\_50MS ; 延时等待转换

ACALL SETUP\_TEMP ; 准备读取

ACALL GET\_TEMP ; 读取温度

ACALL DELAY\_50MS ; 延时

ACALL DELAY\_50MS ; 再延时一次，确保稳定

ACALL SHOW\_KEY\_NUM ; 刷新十位显示

MOV A,B ; 个位移入A

ACALL DELAY\_50MS ; 延时

ACALL SHOW\_KEY\_NUM ; 刷新个位显示

MOV A,#1 ; 光标定位

MOV B,#8

ACALL MOV\_CURSOR

ACALL CMP\_TEMP ; 再次比较温度

JNC CALL\_TRIGGER\_ALARM ; 若仍超限则继续报警

; 继续循环

CONTINUE:

JMP GET\_TEMP\_LOOP ; 无条件跳回GET\_TEMP\_LOOP，持续监测

;-----------------------------------------------------

; 固定字符串："Current:"

SHOW\_CURRENT\_TEMP: DB 43H,75H,72H,72H,65H,6EH,74H,3AH

;=====================================================

; 延时子程序区

; 约50ms延时（使用定时器0模式1）

DELAY\_50MS:

MOV TMOD,#01H ; 定时器0模式1（16位定时器）

MOV TH0,#9EH ; 装载初值高8位

MOV TL0,#5DH ; 装载初值低8位

SETB TR0 ; 启动定时器0

LOOP1:

JNB TF0,LOOP1 ; 等待定时器溢出

CLR TF0 ; 清除溢出标志

CLR TR0 ; 关闭定时器

RET ; 返回

; 约15ms延时（键盘消抖用）

DELAY\_15MS:

MOV TMOD,#01H ; 定时器0模式1

MOV TH0,#0E2H ; 装载初值高8位

MOV TL0,#0B9H ; 装载初值低8位

SETB TR0 ; 启动定时器

LOOP2:

JNB TF0,LOOP2 ; 等待溢出

CLR TF0 ; 清除标志

CLR TR0 ; 关闭定时器

RET ; 返回

; 约1s延时（报警后手动模式用）

DELAY\_1S:

MOV R5,#03H ; 外层循环3次

LOOP\_8:

MOV R6,#0FFH ; 中层循环255次

LOOP\_9:

MOV R7,#0FFH ; 内层循环255次

LOOP\_10:

NOP ; 空操作延时

NOP

NOP

DJNZ R7,LOOP\_10 ; 内层循环

DJNZ R6,LOOP\_9 ; 中层循环

DJNZ R5,LOOP\_8 ; 外层循环

RET ; 返回

; 约60µs延时（DS18B20时序用）

DELAY\_60US:

MOV R3,#10 ; 循环10次

LOOP\_11:

NOP ; 空操作延时

NOP

NOP

NOP

DJNZ R3,LOOP\_11 ; 循环10次约60µs

RET ; 返回

;=====================================================

; LCD初始化子程序：按1602手册典型步骤

SETUP\_LCD:

MOV P2, #1FH ; 初始化P2口为控制线状态

MOV P0, #01H ; 发送清屏指令

SETB P2.7 ; 产生E高脉冲，执行清屏

ACALL DELAY\_15MS ; 延时确保指令完成

MOV P2,#1FH

MOV P0,#06H ; 输入模式：地址+1，显示不移动

SETB P2.7 ; 产生E脉冲

ACALL DELAY\_15MS ; 延时

MOV P2,#1FH

MOV P0,#0EH ; 显示开，光标开，闪烁关

SETB P2.7 ; 产生E脉冲

ACALL DELAY\_15MS ; 延时

MOV P2,#1FH

MOV P0,#38H ; 8位总线，2行显示，5×7点阵

SETB P2.7 ; 产生E脉冲

ACALL DELAY\_15MS ; 延时

CLR P2.7 ; 确保E为低，结束初始化

RET ; 返回

;-----------------------------------------------------

; 显示一个字符子程序

SHOW\_STR:

CLR P2.7 ; 确保RS=0时为写指令，此处RS=1写数据

MOV P0,A ; 将字符数据送P0口

MOV P2,#5FH ; 设置RS=1(数据),RW=0(写),E=1(使能)

SETB P2.7 ; 产生高脉冲，写入数据

ACALL DELAY\_50MS ; 延时确保写入完成

RET ; 返回

; 带额外延时的显示字符，用于键盘回显

SHOW\_KEY\_NUM:

CLR P2.7 ; 确保RS=0

MOV P0,A ; 字符数据送P0

MOV P2,#5FH ; RS=1,RW=0,E=1

SETB P2.7 ; 产生E脉冲

ACALL DELAY\_50MS ; 延时

ACALL DELAY\_50MS ; 额外延时

CLR P2.7 ; 确保E为低

RET ; 返回

; LCD换行到第二行首地址

NEXT\_LINE:

MOV P2,#1FH ; 控制线初始化

MOV P0,#0C0H ; 第二行DDRAM地址指令

SETB P2.7 ; 产生E脉冲

ACALL DELAY\_15MS ; 延时

CLR P2.7 ; 确保E为低

RET ; 返回

; 光标定位子程序：A=0第一行，A=1第二行；B=列号

MOV\_CURSOR:

PUSH ACC ; 保存A寄存器

CJNE A,#0,SECOND\_LINE ; 若A不等于0则跳第二行处理

MOV A,#80H ; 第一行首地址为80H

JMP CALC\_COL ; 跳转到列偏移计算

; 第二行首地址处理

SECOND\_LINE:

MOV A,#0C0H ; 第二行首地址为C0H

; 计算列偏移

CALC\_COL:

ADD A,B ; 行首地址+列偏移得目标地址

MOV P2,#1FH ; 控制线初始化

MOV P0,A ; 将地址指令送P0

SETB P2.7 ; 产生E脉冲

ACALL DELAY\_15MS ; 延时

CLR P2.7 ; 确保E为低

POP ACC ; 恢复A寄存器

RET ; 返回

; 光标左移一位

SHIFT\_CURSOR\_LEFT:

MOV P2,#1FH ; 控制线初始化

MOV P0,#10H ; 光标左移指令

SETB P2.7 ; 产生E脉冲

ACALL DELAY\_15MS ; 延时

CLR P2.7 ; 确保E为低

RET ; 返回

;=====================================================

; DS18B20初始化子程序

SETUP\_TEMP:

CHECK\_EXIST:

CLR P3.7 ; 拉低总线480µs以上，产生复位脉冲

MOV R7,#240 ; 延时计数

DELAY\_LOOP1:

DJNZ R7,DELAY\_LOOP1 ; 循环延时约480µs

SETB P3.7 ; 释放总线

MOV R7,#5 ; 延时计数

ACALL DELAY\_60US ; 延时等待DS18B20响应

JB P3.7,CHECK\_EXIST ; 若无低电平响应则重试

MOV R7,#120 ; 等待存在脉冲结束

WAIT:

DJNZ R7,WAIT ; 延时等待

MOV A,#0CCH ; 发送Skip ROM命令，跳过ROM匹配

ACALL WRITE\_DS18B20 ; 调写入子程序

RET ; 返回

;-----------------------------------------------------

; 向DS18B20写一个字节

WRITE\_DS18B20:

MOV R7,#8 ; 8位数据

COMMAND:

RRC A ; 取最低位到C

CLR P3.7 ; 拉低产生写时隙

NOP ; 短暂延时

NOP

JC WRITE\_ONE ; 若位为1则跳WRITE\_ONE

MOV R6,#28 ; 写0保持低60µs

DELAY\_LOOP3:

DJNZ R6,DELAY\_LOOP3 ; 延时

SJMP BIT\_END ; 跳到结束

; 写1处理

WRITE\_ONE:

SETB P3.7 ; 位为1则在15µs内置高

MOV R6,#27 ; 延时计数

DELAY\_LOOP4:

DJNZ R6,DELAY\_LOOP4 ; 延时

BIT\_END:

SETB P3.7 ; 确保总线释放

NOP

DJNZ R7,COMMAND ; 8位未写完则继续

RET ; 返回

;-----------------------------------------------------

; 启动一次温度转换命令44H

CONVERT\_TO\_DEC:

MOV A,#44H ; 温度转换命令

ACALL WRITE\_DS18B20 ; 发送命令

RET ; 返回

;-----------------------------------------------------

; 读取DS18B20温度寄存器

GET\_TEMP:

MOV A,#0BEH ; 发送读Scratchpad命令BEH

ACALL WRITE\_DS18B20 ; 发送命令

MOV R6,#9 ; 连续读9字节

MOV R0,#70H ; 外部RAM起始地址70H

; 循环读取9字节

READ\_NUM:

READ\_ONE\_BYTE:

MOV R7,#8 ; 每字节8位

READ\_ONE\_BIT:

CLR P3.7 ; 产生读时隙

NOP

NOP

SETB P3.7 ; 释放总线

NOP

MOV C,P3.7 ; 采样总线数据

RRC A ; 移入累加器A

ACALL DELAY\_60US ; 延时60µs

DJNZ R7,READ\_ONE\_BIT ; 8位未读完继续

MOVX @R0,A ; 存外部RAM

INC R0 ; 地址+1

DJNZ R6,READ\_ONE\_BYTE ; 9字节未读完继续

; 以下为把读取的16位温度转成BCD十进制显示

MOV R0,#70H ; 指向温度数据首地址

MOVX A,@R0 ; 读取低字节

ANL A,#0F0H ; 取高4位整数部分

SWAP A ; 高低4位交换

MOV R1,A ; 暂存高4位

MOV R0,#71H ; 指向高字节

MOVX A,@R0 ; 读取高字节

ANL A,#07H ; 取低3位整数部分

SWAP A ; 交换位置

ORL A,R1 ; 合并得8位整数温度值

MOV R0,#7AH ; 暂存地址7AH

MOVX @R0,A ; 保存整数温度值

MOV B,#10 ; 准备除以10

DIV AB ; 拆成十位和个位

ORL A,#30H ; 十位转ASCII

ORL B,#30H ; 个位转ASCII

RET ; 返回，A=十位ASCII，B=个位ASCII

;-----------------------------------------------------

; 将外部RAM中ASCII码转成二进制数值

CONVERT\_ASCII\_TO\_DEC:

MOVX A,@R0 ; 读取ASCII码

ANL A,#0FH ; 取低4位（数字部分）

MOV B,#10 ; 准备乘以10

MUL AB ; 十位×10

MOV B,A ; 结果暂存B

INC R0 ; 指向下一位

MOVX A,@R0 ; 读取个位ASCII

ANL A,#0FH ; 取数字部分

ADD A,B ; 十位+个位得二进制阈值

RET ; 返回

;-----------------------------------------------------

; 比较当前温度与设定阈值（结果：C=1表示当前温度≥阈值）

CMP\_TEMP:

MOV R0,#30H ; 指向阈值存储地址30H

ACALL CONVERT\_ASCII\_TO\_DEC ; 阈值→A（二进制）

MOV B,A ; 阈值暂存B

MOV R0,#7AH ; 指向当前温度存储地址7AH

MOVX A,@R0 ; 读取当前温度（二进制）

CJNE A,B,ok ; 比较温度与阈值

ok:

RET ; 返回，C=1表示当前≥阈值

;=====================================================

; 报警处理子程序

TRIGGER\_ALARM:

MOV R5,#12 ; 声光报警循环12次

CLR P2.7 ; 确保LED初始状态

; 报警循环

ALARM\_LOOP:

SETB P1.0 ; 启动风扇（无论自动/手动模式）

CLR P2.0 ; LED亮

CLR P2.5 ; 蜂鸣器响

ACALL DELAY\_50MS ; 延时50ms

SETB P2.0 ; LED灭

SETB P2.5 ; 蜂鸣器停

ACALL DELAY\_50MS ; 延时50ms

DJNZ R5,ALARM\_LOOP ; 12次未结束则继续

; 判断是否为手动模式

JB 20H.0,MANUAL\_ALARM\_HANDLE ; 若为手动模式则跳手动处理

; 自动模式：直接返回，关闭风扇

CLR P1.0 ; 关闭风扇

RET ; 返回

;-----------------------------------------------------

; 手动模式：等待用户按键解除报警

MANUAL\_ALARM\_HANDLE:

ACALL CHECK\_KEY\_PRESS ; 检查是否有键按下

JC ALARM\_CLEARED ; 若有键按下则跳ALARM\_CLEARED

; 无键继续报警

MOV R5,#12 ; 重置循环次数

SJMP ALARM\_LOOP ; 跳回报警循环

; 手动停止报警，延时5秒

ALARM\_CLEARED:

ACALL DELAY\_1S ; 延时1秒

ACALL DELAY\_1S

ACALL DELAY\_1S

ACALL DELAY\_1S

ACALL DELAY\_1S ; 共延时5秒

CLR P1.0 ; 关闭风扇

RET ; 返回

;=====================================================

; 检查是否有按键按下子程序

; 返回：C=1表示确认有键按下

CHECK\_KEY\_PRESS:

MOV P1,#0FH ; 设置P1口低4位输入

MOV A,P1 ; 读取P1状态

ANL A,#0FH ; 保留低4位

CJNE A,#0FH,KEY\_PRESSED ; 若不全为1则有键按下

CLR C ; 无键按下，C=0

RET ; 返回

; 消抖确认

KEY\_PRESSED:

ACALL DELAY\_15MS ; 延时消抖15ms

MOV P1,#0FH ; 再次设置

MOV A,P1 ; 再次读取

ANL A,#0FH ; 保留低4位

CJNE A,#0FH,KEY\_CONFIRMED ; 确认仍有键按下

CLR C ; 抖动，C=0

RET ; 返回

; 确认有键按下

KEY\_CONFIRMED:

SETB C ; 确认有键按下，C=1

RET ; 返回

END ; 程序结束

参考文献

[1]张斌,孙宇,陈琳.基于STC单片机的温度报警器研究[J].山西电子技术,2024,(04):118-120.

[2]柳文静.基于单片机的温度报警器[J].电子测试,2020,(03):5-7.DOI:10.16520/j.cnki.1000-8519.2020.03.001.

[3]陈洁鉴,吴建文.基于单片机的温度监控系统[J].电子元器件与信息技术,2019,(02):25-30.DOI:10.19772/j.cnki.2096-4455.2019.2.007.