用单片机实现温度远程显示

*[作者：刘楚浩 *发表于《单片机与嵌入式系统应用》2002 年第 9 期]*

摘 要：文章介绍了用 AT89S8252 单片机的串行接口与智能温度巡回检测仪（XJ-08S）通过 RS—485 总线相互通讯实现热水温度远程显示的一种低成本解决方案，内容涉及 RS—485 总线通讯、单片机驱动数码管显示、数据转换以及键盘处理软硬件设计等内容。

关键词：单片机 RS—485 总线 数码管显示 数据转换 键盘处理

一、前 言

目前检测温度一般采用热电偶或热敏电阻作为传感器，这种传感器至仪表之间一般都要用专用的温度补偿导线，而温度补偿导线价格很贵，并且线路太长也会影响测量精度。在实际应用中往往需要对较远处（1KM 左右）的温度信号进行监视。现有的解决方案有很多，例如：

1、 在现场用智能仪表对温度信号进行测量，用计算机作上位机与智能仪表进行通讯来实现远程温度监测（采用这种方案要增加计算机设备及相关计算机软件）。

2、 NCU+DDC 实现远程温度监测。用两个 DDC，一个安装在现场测量温度，另一个安装在监视地，两个 DDC 通过 NCU 进行通讯从而实现远程温度监测。

但以上方案都存在成本高的问题，有没有低成本的解决方案呢？其实，在单片机应用日益广泛的今天，完全可以用单片机以极低的成本来实现远程温度监测。

二、问题的提出

我单位管理的锅炉房同时给两栋建筑物内的两家酒店供应蒸汽，由安装在两栋建筑物地下室的热交换器进行热交换后产生热水送给客房。从锅炉房至两个热交换站的距离分别约 600 米，值班人员要不停地奔波于两个热交换站与锅炉房之间进行设备巡视，检查热水温度是否控制在规定的范围，这样不仅增加了值班人员的劳动强度，同时也使锅炉房经常无人(因每班 1 人值班)。如果能在锅炉房显示两个热交换站内各热交换器的热水温度，则值班人员仅在热水温度异常时才需到各热交换站检查设备，这样便可解决上述问题。我公司曾就此问题找专业公司作过方案，其报价在人民币１０万元左右，后因种种原因该项目未实施。经过分析，本人发现可以用单片机+智能仪表以低成本实现温度远程显示，并且经过实验取得了成功，现将设计方案简述如下：

三、控制要求及解决方案选择

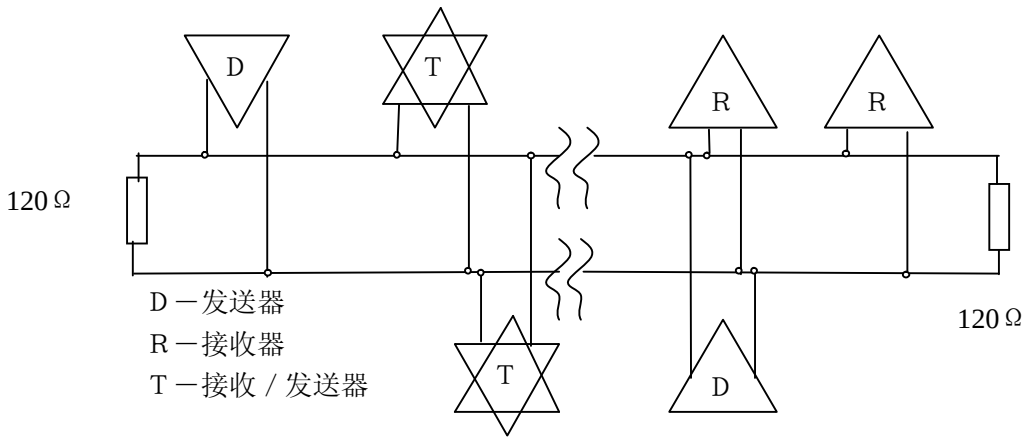
1、 两个热交换站分高低区共安装有 8 个热交换器，正常水温在 45℃至 65℃之间；两个热交换站与锅炉房的距离分别为 500 米和 600 米左右。

2、 要求在锅炉房能以巡回及定点两种方式显示 8 个热交换器的热水温度，巡回方式以 3 秒为周期轮流更新及显示各热交换器热水温度。定点方式时每按上键或下键一次则显示上或下一个热交换器热水温度，每 3 秒自动更新数据一次。

3、 根据控制要求选择单片机+智能仪表的解决方案：用带通讯接口的智能仪表安装在现场测量温度，设计制作一个单片机装置完成与智能仪表的通讯及数据显示。

四、通讯协议、智能仪表选择及其参数介绍

因热水温度信号变化较慢，因而对通信的速度要求不高，对于这种低速率远距离的通讯选用 RS-485 总线适宜。RS-485 是 EIA（美国电子工业联合会）在 1983 年公布的新的平衡传输标准，是工业界使用最为广泛的双向、平衡传输线标准接口，它以半双工方式通信，支持多点连接，传统驱动器允许创建多达 32 个节点的网络，且其具有传输距离远（最大传输距离为 1200M），传输速度快（1200M 时为 100KBPS）等优点。其连接方法如下图所示。



为了满足现场温度检测及与单片机装置通讯的要求，必须选择至少有 5 个温度检测点及具有 RS—485 通讯端口的智能仪表。经过对市场上常用的温度检测仪进行分析，选择由重庆川仪十八厂生产的 XJ-08S 型巡回检测仪作现场测量仪表。

（一）该仪表主要特点如下：

1、 多量程方式，热电偶、热电阻，1-5V 标准信号混合输入，可通过键盘进行设置；

2、 最多 8 个测量通道（能测量 8 个温度信号）；

3、 采用 RS-485 通讯标准，可将各通道最新数据向上位机传送。

重要的是，该仪表的说明书详细介绍了与该仪表进行数据交换的命令及格式，其通讯协议也相对较简单,这给我们用单片机实现温度远程显示降减低了难度(虽然有 RS-485 端口的仪表很多，但大多没有通讯命令的详细说明，给我们用单片机编程增加了难度)。

（二）XJ-08S 巡回检测仪通讯协议

1、通讯口设置

- 通讯方式：RS-485 标准电平
- 同步方式：起停同步方式
- 波特率：9600BPS
- 通讯距离：不大于 1200M
- 通讯线：二线
- 数据代码：ASCII 码
- 数据格式：每字符 10 位，1 个起始位，8 个数据位，1 个停止位

2、 数据传输格式

- 地址：2 字节（高字节在前，低字节在后）;
- 数据：按地址顺序，仪表数据传输格式为十六进制 2 字节定点数;

  2 字节定点数=低字节高 4 位（ASCII 码）+低字节低 4 位（ASCII 码）

  高字节高 4 位（ASCII 码）+高字节低 4 位（ASCII 码）

  若数据为负数，则采用补码方式传输;
- 在传输实时测量值时，传输完 2 字节定点数后，紧接着又传输 2 字节定点数，其中高字节低 4 位为小数点位数。

例 ：$(50.0)_{10}$ 表示为 46 34 30 31 30 30 30 31

低字节 高字节 小数位数

3、 仪表通讯格式：

@ DE 帧类型 帧数据 CRC CR

- @ ：通讯起始符
- DE ：仪表设备号（地址）
- 帧类型 ：操作命令
- 帧数据 ：各种操作命令所对应的命令及数据
- CRC ：校验字节（除@外 CRC 之前的其他几个字节的异或值）
- CR ：结束符（回车符）

4、应用中用到的命令及数据格式：

- 读仪表全部动态数据命令帧 @ DE RD CRC CR
- 命令回送帧 正确 @DE RD 帧数据 CRC CR

  错误 @DE * * CRC CR

例：读 28 号仪表的全部动态数据

命令：’@1CRD64’,0D(ASCII 码 40 31 43 52 44 36 34 0d)

错误返回码 ‘@1C**72’,0D(ASCII 码 40 31 43 2A 2A 37 32 0D)

正确返回数据’@ 1C RD XXXX XXXX XXXX XXXX XXXX XXXX

第 0 通道 第 1 通道 第 2 通道

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

第 3 通道 第 4 通道 第 5 通道 第 6 通道

XXXX XXXX XX’,0D

第 7 通道 校验

五、单片机选择及硬件电路设计

1、 选用 ATMEL 公司生产的 AT89S8252-24PC 单片机，其主要参数及特点如下：

- 与 MCS-51 产品兼容（其引脚图见原理图）
- 具有 8K 字节可擦写的 FLASH 内部程序存储器，可擦写 1000 次；2K 字节 EEPROM，可擦写 100，000 次，SPI 口（用 PC 机的并口连接 5 条线即可通过 SPI 口下载程序，下载软件可从网上下载，这样可节省购买编程器的费用；）。

  注：笔者现已制作成了 ISP2000 三合一烧写器，详情见 http://www.mcudiy.com
- 256 字节 RAM，32 根可编程 I/O 线，可编程串行口，内置看门狗。

  与看门狗有关的特殊功能寄存器 WMCON 地址= 96H，与看门狗有关的控制位为 96h 第 0、1、5、6、7 位，第 5、6、7 位用于设置看门狗定时时间（具体见第 5 页表格），本应用中第 5、6、7 位均置 1，设置看门狗溢出时间为 2048ms，第 0 位为看门狗使能控制位，该位置 1 将使能看门狗，其第 1 位为复用位，向第 1 位写 1 将复位看门狗定时器，具体操作如下：

  a,使能看门狗，并将其溢出时间设定为 2048ms：ORL 96H, #0E1H;

  b,看门狗定时器清 0：ORL 96H, #2

2、按键设计：

为方便使用，设计了三个按键，分别为巡回/定点切换键、上键、下键。切换键用于巡检与定点模式的切换，上键向上切换通道，下键向下切换通道；其中巡检/定点切换键通过外部中断 1 以中断方式工作，中断程序将巡回/定点标志取反后直接跳到主程序中巡回/定点标志判断程序前运行，由判断程序完成巡回/定点的切换。按键信号由单片机 P3.3，P3.4，P3.5 引脚输入。

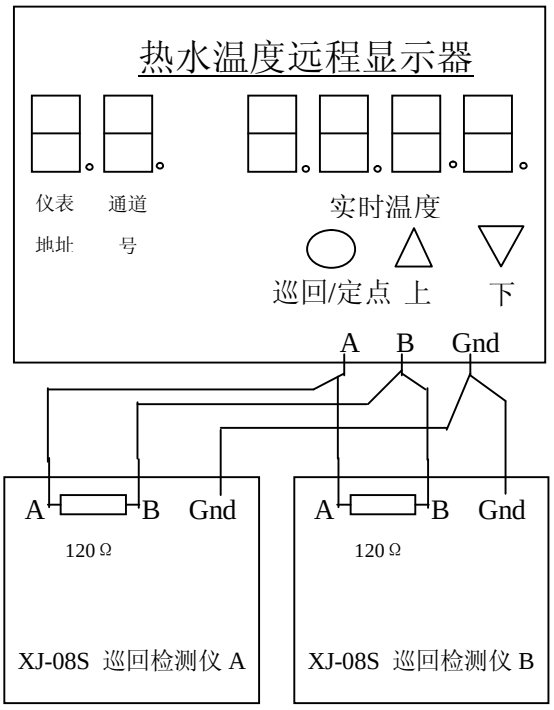
3、显示电路设计

为方便观察，选用三个二位共阳级 8 段数码显示管（TOD5201AE）动态显示，一位显示仪表地址（从 A 至 F），一位显示通道号（从 0 至 7 通道），其余四位用来显示实时温度值;用单片机 P1 口驱动一片 74HC244 以吸收电流的方式控制段码,用单片机 P0.0 至 P0.5 引脚驱动六个 P N P 三极管（9012）控制位选。

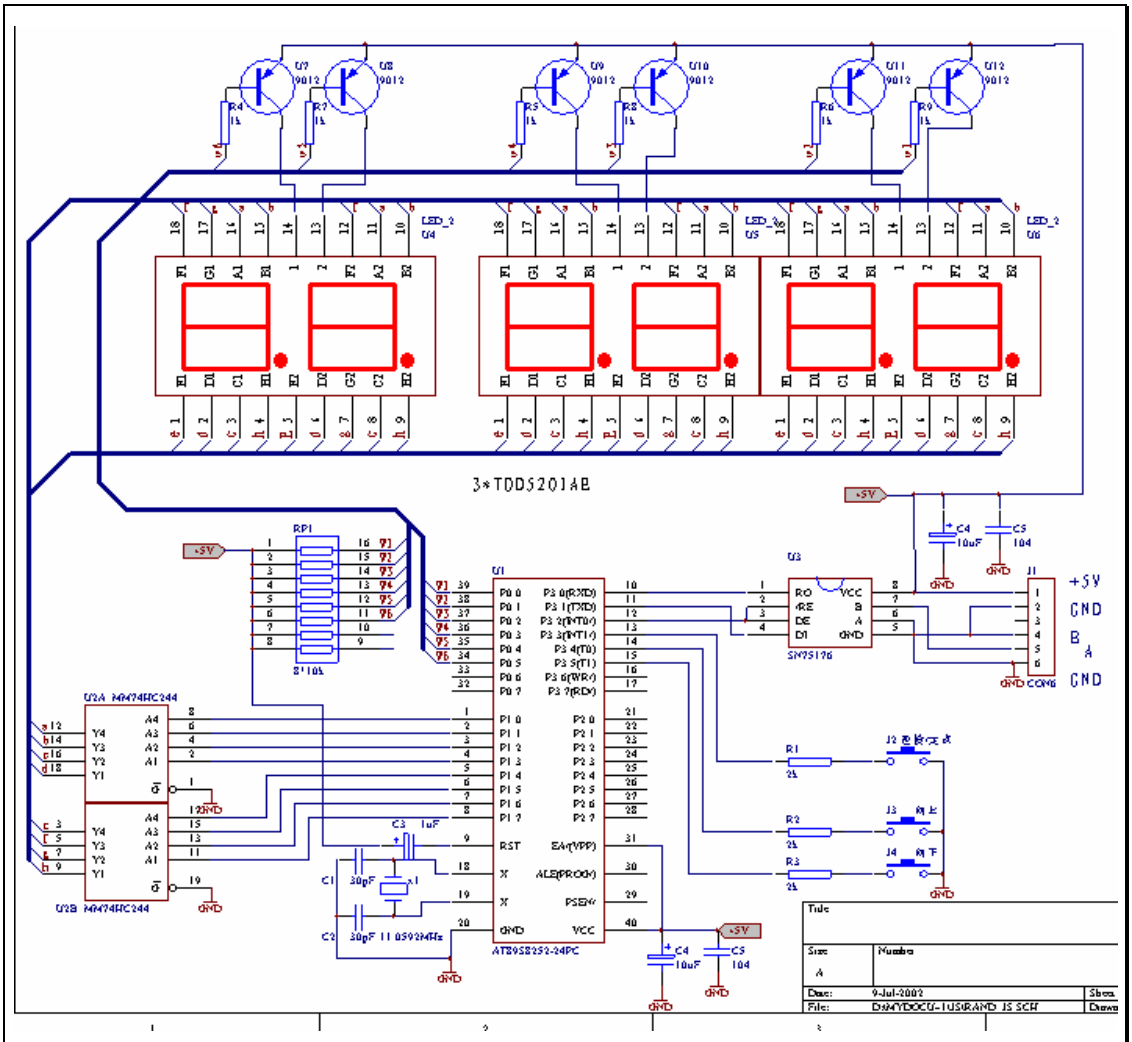
4、电源选用 5V/1A 市售成品开关电源。

5、 根据智能仪表通讯协议的要求，串行口定义为方式 1 工作，一帧 10 位：1 个起始位、8 个数据位、一个停止位；用一片 75176 完成数据的发送与接收，由于 RS-485 为半双工，故用单片机 P3.2 引脚控制发送与接收的切换；两个智能仪表处于 RS-485 总线的两个端点，为提高可靠性，在 RS-485 总线的两个端点上分别并联一个 120Ω、1/4W 终端电阻。

- 系统方框图如右示：
- 单片机装置电原理图见下页：



看门狗溢出时间表			
7	6	5	定时值
0	0	0	16ms
0	0	1	32ms
0	1	0	64ms
0	1	1	128ms
1	0	0	256ms
1	0	1	512ms
1	1	0	1024ms
1	1	1	2048ms


3*TDD5201AB
+5V
GND
B
A
GND
Title
Size
A
Number
Date:
9-Jul-2002
File:
D:\MYDOCU~1\USB\RAND_IS.SCH

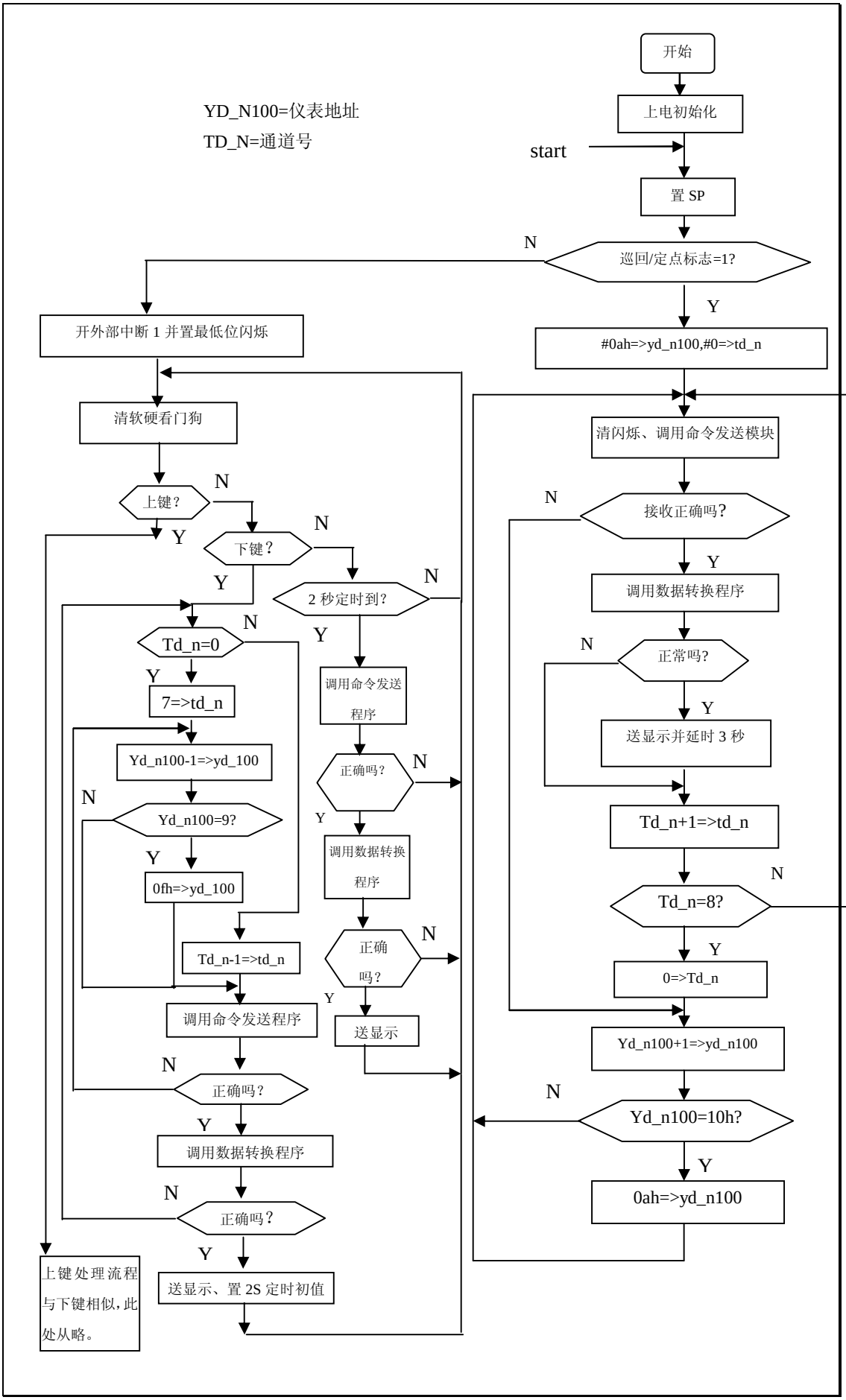
六、单片机软件设计说明及程序流程图

为方便调试，采用模块化编程，共分 1 个主模块及 5 个子模块，各模块功能说明及流程图如下（源程序见附录）：

1、主模块（rand_main.a51）

上电后程序显示初始化标志并等待 3 秒，然后进入巡回模式，自动检测地址为 A（10）至 F（15）的智能仪表，对在线的仪表及已使用通道以 3 秒为周期自动巡回读取实时温度并送数码管显示，仪表未在线或未用通道则自动跳过；定点模式时根据上、下按键情况自动向前或向后移动一个通道后等待下一次按键（向上至最高通道号时跳到上一个仪表的最低在用通道，向下至最低通道号时跳到下一个仪表的最高在用通道）；等待期间以 3 秒为周期读取当前仪表当前通道实时温度并送数码管显示，另外，为了区分巡回、定点工作模式，定点工作模式时最低位数码管以约 0.5Hz 的频率闪烁显示。当检测到巡回/定点切换键按下时自动在巡回及定点模式间转换，当检测到温度值低于－199.9 度时显示-199.9 度。

主模块流程图见下页：


YD_N100=仪表地址
TD_N=通道号
开始
上电初始化
start
置 SP
巡回/定点标志=1?
N
Y
#0ah=>yd_n100,#0=>td_n
清闪烁、调用命令发送模块
接收正确吗?
调用数据转换程序
正常吗?
送显示并延时 3 秒
Td_n+1=>td_n
Td_n=8?
0=>Td_n
Yd_n100+1=>yd_n100
Yd_n100=10h?
0ah=>yd_n100
开外部中断 1 并置最低位闪烁
清软硬看门狗
上键?
下键?
2 秒定时到?
Td_n=0
7=>td_n
Yd_n100-1=>yd_100
Yd_n100=9?
0fh=>yd_100
Td_n-1=>td_n
调用命令发送程序
正确吗?
调用数据转换程序
正确吗?
送显示、置 2S 定时初值
调用命令发送程序
正确吗?
调用数据转换程序
正确吗?
送显示
上键处理流程与下键相似,此处从略。

2、数据转换子模块（rand_data.a51）

功能： 本模块先将 ASCII 码转换成 BIN 码，然后将 BIN 码转换成 BCD 码并将数据转换成可直接显示的格式

入口： a=通道号

出口： r1=个位,r2=十位,r3=百位,r4=千位（显示数据）

a=非 0 表示该通道未用

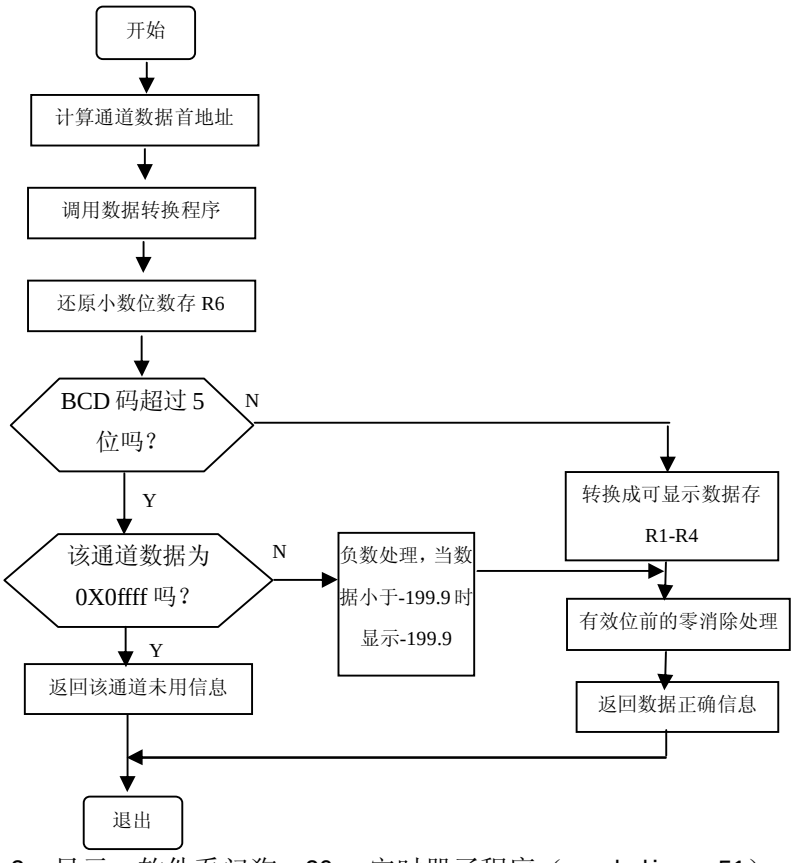
注：本子模块中直接将小数点信息加到相应位数据上，如需在某位显示小数点，则将该位数据加 10H，显示子模块根据此信息显示小数点。

各子程序功能介绍：

- bcd2_bcd1：将 r3,r4 中的 4 位 bcd 码由低至高分别存入 r1,r2,r3,r4 中；
- ascii4_bcd3:将(r0)为首地址的 4 字节 ascii 码转换成 bcd 码并将 bcd 码由低至高依次存放在 r3,r4,r5 中，r3,r4 中各存放 2 位 bcd 码，r5 中存放一位 bcd 码；
- bin_bcd:将 r1,r2 中的 bin 码转换成 bcd 码并将 bcd 码由低至高依次存放在 r3,r4,r5 中，r3,r4 中各存放 2 位 bcd 码，r5 中存放一位 bcd 码；
- ascii4_bin2：将(r0)为首地址的 4 字节 ascii 码转换成 2 位 bin 码存放在 r1（低字节）,r2(高字节）中；

  注：[（r0）指 r0 中内容]
- ascii_bin：将（r0）为地址的一位 ascii 码转换成 bin 码存 A 中。

（高级子程序调用低级子程序以实现更强的功能，源程序中对各子程序有较详细的注释，故此处仅画出主流程图）

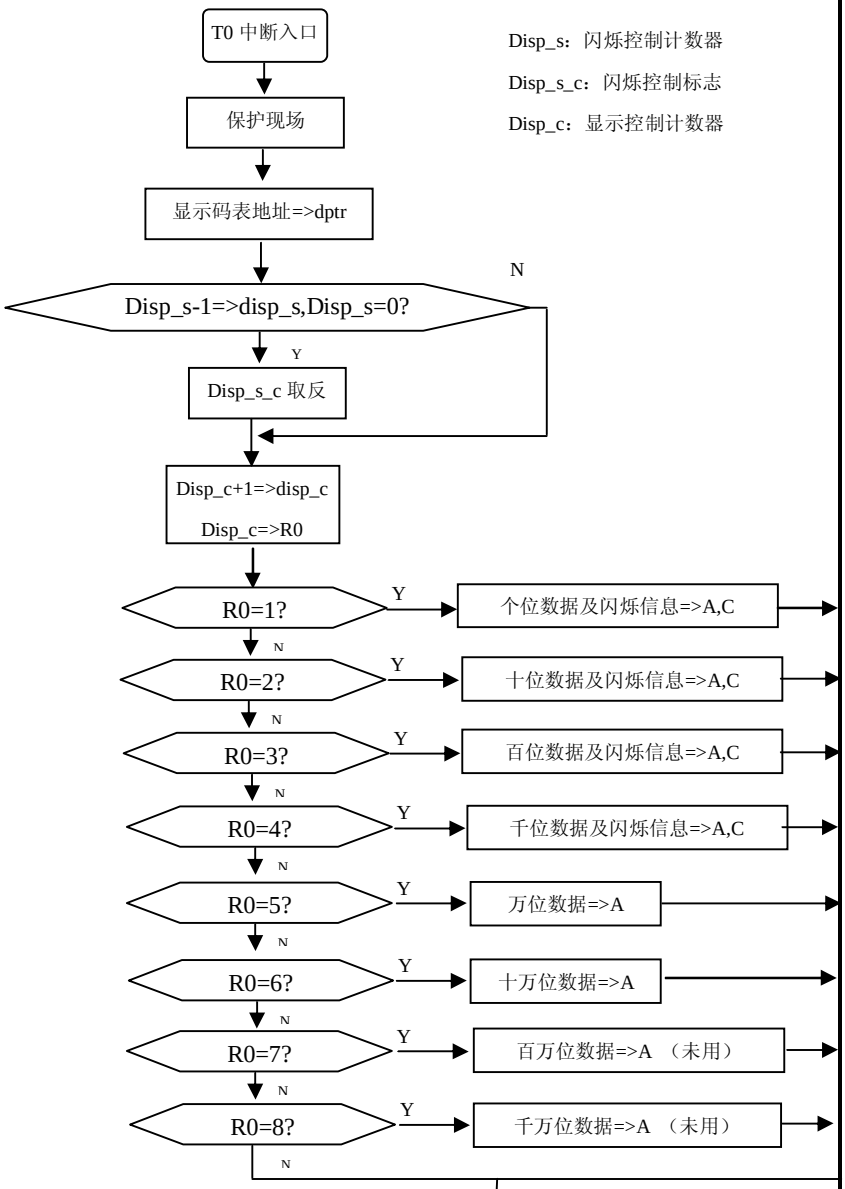


3、显示、软件看门狗、20ms 定时器子程序（rand_disp.a51）

功能： 通过定时器 0 以 1MS 为周期动态显示缓冲区中的内容；同时完成 20ms 定时器及软件看门狗计数；当软件看门狗定时器溢出时自动复位。

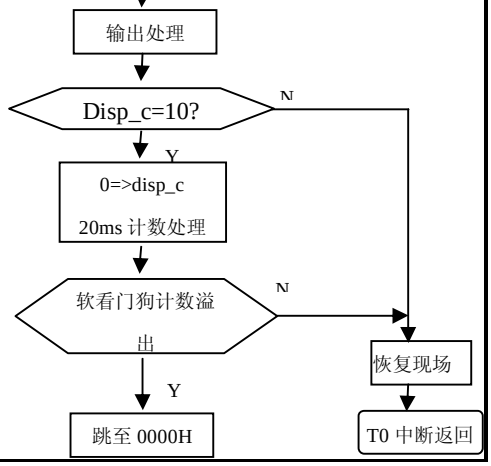
入口： 以 disp_buf 为首地址依次存放从个位至十万位待显示数据，以 disp_wc 为首地址（位地址）依次存放个位至千位闪烁控制位信息，为'0'常亮，为'1'闪烁.

出口： wdt 中存放软件看门狗计数值，time_20ms 中存放 20ms 的计数值。Wdt 溢出时跳至程序存储器 0000H 单元



- 将定时器 0 设定为模式 1 定时工作，每 ms 产生一次中断，即显示一位，每 10ms 6 位数据轮流显示 1ms；为进一步提高可靠性，在该模块中设计了软件看门狗，实现方法见源程序。

  模块流程图如本页所示：



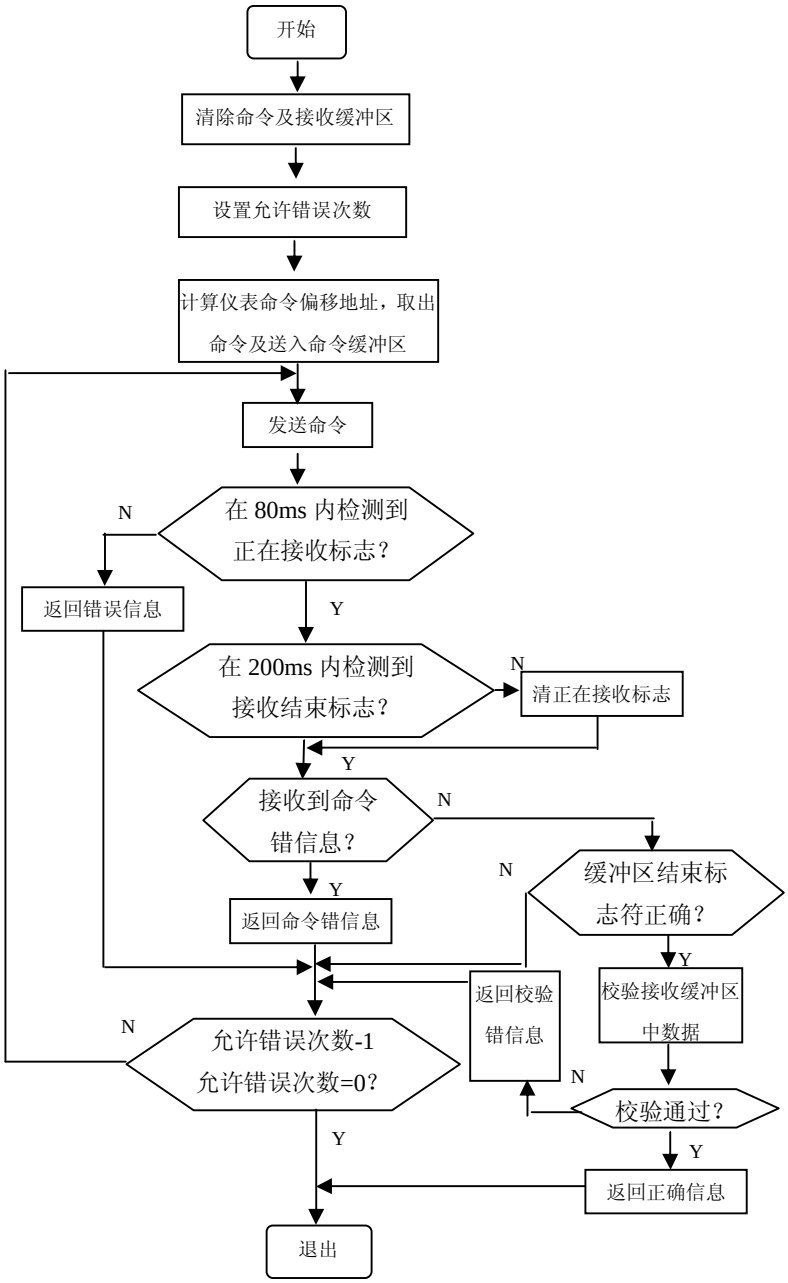
4、命令发送及接收控制子模块（rand_send.a51）

功能： 发送命令缓冲区中命令并等待接收完毕,然后检查接收到的数据,接收错误则重发命令,达到设定的次数仍错则存入错误码后退出.

入口： A=仪表地址

出口： A=0 表示接收正确 1 表示校验错 '*'表示命令错 'N'表示无回应

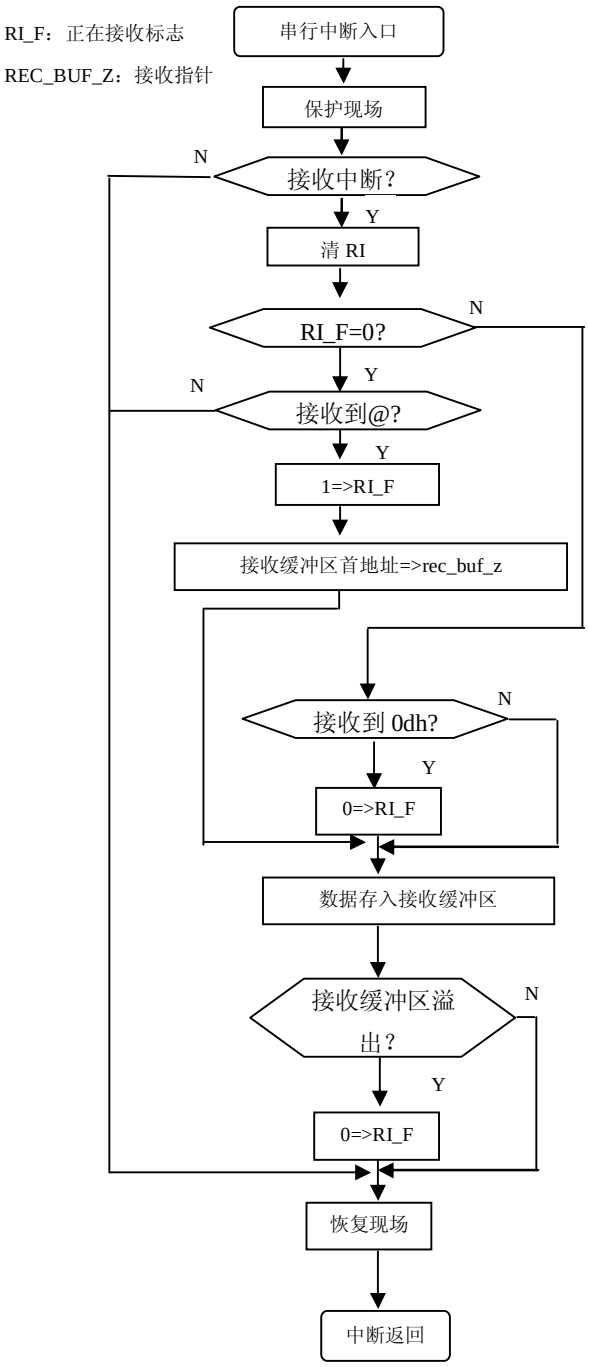
模块流程图如下：



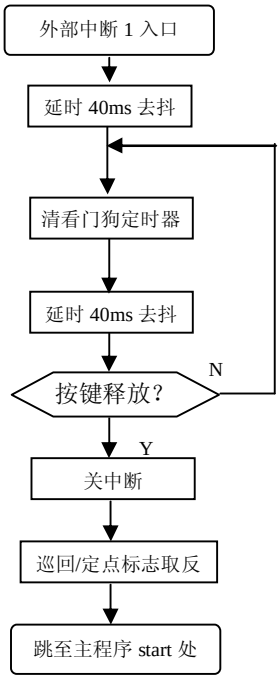
5、串行接收子模块（rand_rec.a51）

功能： 串行接收模块通过中断自动执行，程序检测到@时置 ri_f 标志并开始接收，检测到回车符时清 ri_f 后退出,接收到的数据存放在以 rec_buf 为首地址的缓冲区中。

串行中断（rand_rec.a51）流程图



巡回/定点键中断(rand_key.a51)流程图



6、巡回／定点转换检测模块（rand_key.a51）

功能： 模块通过外部中断 1 自动执行，检测到巡回／定点按键后先延时去抖动，并等待按键释放后将巡回／定点标志位取反，然后自动跳至主模块中 start 处，并不返回原中断点。

模块流程图如见本页：

七、调测及运行情况介绍

1、 调试步骤：

- 硬件连接：将各芯片插在面包板上并用导线按原理图连接
- 软件开发环境：Keil uVision2 for Windows ，该软件的模拟调试器支持 C 语言及汇编语言源代码调试，其汇编程序支持宏汇编及模块化编程，使用方便。
- 首先在模拟调试器中调试各子模块，各子模块调试正常后再将各子模块及主模块全部汇编连接成目标文件，最后将目标代码下载到单片机中并连接智能仪表进行统调。

2、 该系统软件部分除报警子模块外已全部开发完成并实际测试通过，测试时用一对长度约 1100M 的 0.5mm^2 普通电话线作为单片机装置与两个智能仪表的通讯线，经过一个多星期的连续运行观察，运行可靠稳定，完全满足使用要求。

[注：报警子模块的开发及硬件制版工作正在进行中。]

八、结束语

用单片机+智能仪表构成的温度远程显示系统总造价约人民币 7000 元（两块智能仪表+温度传感器约 3500 元，通讯电缆约 2000 元，单片机装置及施工费用约 1500 元），与采用其它方案的系统比较造价较低，其硬件及软件的设计、制作都较简单，有一定的电子及单片机知识便可完成。同时对该装置软件稍加修改即可与其他带有 RS-485 通讯端口的仪器仪表通讯（如变频器、电路传感器等），实现远程测量、监视及控制。

与发达国家相比，我国单片机应用的普及程度较低。通过该实例说明，我们完全可以把我们学到的单片机知识应用到实际工作中，直接创造经济效益。

九、参考资料

<<单片机高级语言 C51Windows 环境编程与应用>> 电子工业出版社
徐爱钧 彭秀华 编著
<<ATMEL89 系列 Flash 单片机原理与应用>> 电子工业出版社
余永权 主编
<<标准集成电路数据手册高速 CMOS 电路>> 电子工业出版社
<<ATMEL AT89S8252 数据手册>> ATMEL 公司
<<XJ-08H(S)巡回检测仪使用说明书>> 重庆川仪十八厂
<<工业控制计算机组成原理>> 清华大学出版社
孙廷才 王杰 孙中健 编著
<<电子技术>>2001 年 11 期

十、附录（源程序）

1、主模块（rand_main.a51）

```
;------------1--------rand_main.a51----------------
name rand_main
EXTRN CODE(I_T0,send_order,rec_si,ascii_bcd,key)
extrn data (DISP_buf,disp_wc,time_20ms,wdt)
extrn idata (rec_buf)
public TH0_H,TL0_L,START,ri_f,wdt_time,key_work,key_f
prog SEGMENT CODE           ;定义代码段
byte_buff segment data      ;定义字节数据段 1
byte_buffI segment idata    ;定义字节数据段 2
bit_buff segment data bitaddressable ;定义位数据段
bit_flag segment bit
stack segment idata         ;定义堆栈数据段
     rseg stack         ;为堆栈保留 20 个字节
     ds 20
     CSEG AT 00H        ;定义绝对代码段
     USING 0            ;选择工作寄存器组第 0 组
     lJMP rand          ;复位时程序执行执行点
     CSEG AT 03H        ;外部中断 0
     RETI
     CSEG AT 0BH        ;定时器 0 中断
     lJMP I_T0
     CSEG AT 13H        ;外部中断 1
     ljmp key
     RETI
     CSEG AT 1BH        ;定时器 1 中断
     RETI
     CSEG AT 23H        ;串行中断
     ljmp rec_si
     RSEG BYTE_BUFF
     td_temp: ds 1
     TD_n: DS 1         ;通道号
     YD_n100: ds 1      ;仪表地址
;---------------------------------
     rseg bit_flag
     key_work:dbit 1
     ri_f: dbit 1       ;串行口正在接收标志
;----主程序
     RSEG prog
;-----------------------------符号定义
     key_f equ p3.3
     key_up equ p3.4
     key_down equ p3.5
     wdt_time equ 100        ;软件看门狗定时器计数值
     scon_c equ 01010000b    ;串行口设置为 10 位 UART,允许接收
     th1_h equ 0fdh          ;晶振为 11.0592M,波特率为 9600 时初值
     tmod_n equ 21h          ;设置定时器（T1 作为波特率发生器工作在模式 2，T0 为模式 1）
     ie_n equ 82h            ;设置串行,定时器 0 中断
     ip_n equ 12h            ;设置串行,定时器 0 中断为高级中断
     TH0_H EQU 0FCH
```

```
    TL0_L EQU 06BH          ;设定定时器 0 为 1MS 中断初值
;------------------初始化代码
    clear_wdt macro             ;清硬看门狗宏定义
    orl 96h,#2
    ENDM
rand:    ORL 96H,#0E1H      ;启动硬件看门狗
    mov p0,#0ffh
    mov r0,#20h
    mov a,#0
start_300:mov @r0,a
    inc r0
    cjne r0,#128,start_300
    setb key_work      ;置巡回 / 定点标志
    mov p3,#0ffh
    mov p2,#0ffh
    mov disp_wc,#0          ;初始化闪烁控制位
    mov ip,#ip_n       ;设置串行,定时器 0 中断为高级中断
    mov scon,#scon_c
    MOV TMOD,#tmod_n
    MOV TH0,#TH0_H          ;设定定时器初值
    MOV TL0,#TL0_L
    mov th1,#th1_h
    mov tl1,#th1_h
    mov ie,#ie_n
    mov pcon,#0
    SETB tr1
    SETB TR0
;-------显示初始化提示
    acall chuhao
    CLEAR_WDT
start:   clr ea
    mov sp,#stack-1
    setb ea
    clr ri_f       ;清接收中标志
    clr p3.2       ;置 75176 为接收状态
    jnb key_work,db_work
;----------------------------------
    setb ex1
    mov yd_n100,#0ah
    mov td_n,#0
s_0: anl disp_wc,#0f0h
    mov wdt,#0          ;清看门狗定时器
    clear_wdt
    mov a,yd_n100
    lcall send_order
    JNZ s_1
    mov disp_buf+5,yd_n100
    mov a,td_n
    lcall ascii_bcd
    jnz s_2
    mov disp_buf+4,td_n
    acall mov_to_disp
    mov wdt,#0
    clear_wdt
    mov a,#75
    add a,time_20ms
    cjne a,time_20ms,$
    mov wdt,#0
```

```
     clear_wdt
     mov a,#75
     add a,time_20ms
     cjne a,time_20ms,$
s_2: inc td_n
     mov a,td_n
     cjne a,#8,s_0
     mov td_n,#0
s_1: inc yd_n100
     MOV A,YD_N100
     cjne a,#10h,s_0
     mov yd_n100,#0ah
     ajmp s_0
;----------------------------
db_work:setb ex1
     orl disp_wc,#01h
d_3: mov wdt,#0          ;清看门狗定时器
     clear_wdt
     jb key_up,d_4
     mov a,#2
     add a,time_20ms
     cjne a,time_20ms,$
     jb key_up,d_4
d_5: mov wdt,#0
     clear_wdt
     jnb key_up,d_5          ;等待向上键松开
;-------------------------------------------------
d_53:    mov a,td_n
     cjne a,#7,d_50
     mov td_n,#0
d_52:    inc yd_n100
     mov a,yd_n100
     cjne a,#10h,d_51
     mov yd_n100,#0ah
     ajmp d_51
d_50:    inc td_n
d_51:    mov a,yd_n100
     lcall send_order
     JNZ d_52
     mov disp_buf+5,yd_n100
     mov a,td_n
     lcall ascii_bcd
     jnz d_53
     mov disp_buf+4,td_n
     acall mov_to_disp
     mov a,#100          ;2 秒钟定时设定
     add a,time_20ms
     ajmp d_3
;-------------------------------------------------
d_4: jb key_down,d_10
     mov a,#2
     add a,time_20ms
     cjne a,time_20ms,$
     jb key_down,d_3
d_6: mov wdt,#0
     clear_wdt
     jnb key_down,d_6   ;等待向下键松开
;-------------------------------------------
```

```
d_43:    mov a,td_n
     cjne a,#0,d_40
     mov td_n,#7
d_42:    dec yd_n100
     mov a,yd_n100
     cjne a,#9h,d_41
     mov yd_n100,#0fh
     ajmp d_41
d_40:    dec td_n
d_41:    mov a,yd_n100
     lcall send_order
     JNZ d_42
     mov disp_buf+5,yd_n100
     mov a,td_n
     lcall ascii_bcd
     jnz d_43
     mov disp_buf+4,td_n
     acall mov_to_disp
     mov a,#100          ;2 秒钟定时设定
     add a,time_20ms
     ajmp d_3
;---------------------------------
d_10:    cjne a,time_20ms,d_30
     mov a,yd_n100
     lcall send_order
     jnz d_30
     mov a,td_n
     lcall ascii_bcd
     jnz d_30
     acall mov_to_disp
     mov a,#100
     add a,time_20ms
d_30:ajmp d_3
;---------------------------------
mov_to_disp:mov disp_buf,r1
     mov disp_buf+1,r2
     mov disp_buf+2,r3
     mov disp_buf+3,r4
     ret
chuhao:   mov r1,#1eh          ;初始化等待提示
     mov r2,#22h
     mov r3,#21h
     mov r4,#1fh
     mov disp_buf+4,#21h
     mov disp_buf+5,#20h
     acall mov_to_disp
     mov a,#75
     add a,time_20ms
     cjne a,time_20ms,$
     mov wdt,#0
     clear_wdt
     mov a,#75
     add a,time_20ms
     cjne a,time_20ms,$
     ret
     end
```

2、数据转换子模块（rand_data.a51）

```
;-----------2-------rand_data.a51----------------
     name   rand_data
extrn idata (rec_buf)
public ascii_bcd
prog SEGMENT CODE           ;定义代码段
byte_buff segment data      ;定义字节数据段 1
byte_buffl segment idata    ;定义字节数据段 2
bit_buff segment data bitaddressable ;定义位寻址区字节数据段
     rseg prog
     using 0
;-------------------------------------
ASCII_BCD:mov b,#8      ;计算该通道数据偏移首地址
     mul ab
     add a,#5
     mov r0,#rec_buf
     add a,r0
     mov b,a
     mov r0,a
     acall ascii4_bcd3
     mov a,b            ;还原小数位数
     add a,#7
     mov r0,a
     acall ascii_bin
     mov r6,a           ;小数位数暂存在 R6 中
     mov a,r5
     jz asc_oK
     mov a,dpl
     cpl a
     clr c
     inc a
     mov dpl,a
     mov a,dph
     cpl a
     addc a,#0
     mov dph,a
     jnz asc_b
asc_0:    mov a,dpl
     dec a
     jnz asc_b
     mov r1,#0fh
     mov r2,#0fh
     mov r3,#0fh
     mov r4,#0fh
     mov a,#0fh
     ajmp asc_out
asc_b:    mov r1,dpl         ;为负数处理
     mov r2,dph
     acall bin_bcd
     mov a,b            ;还原小数位数
     add a,#7
     mov r0,a
     acall ascii_bin
     mov r6,a           ;小数位数暂存 r6
     acall bcd2_bcd1
     mov a,r4
```

```
     dec a
     jnz asc_100         ;数据小于-199.9 时显示-199.9
     mov r4,#23h
     ajmp asc_sw
asc_100:mov r4,#23h
     mov r3,#9
     mov r2,#19h
     mov r1,#9
     ajmp asc_sw_1
asc_ok:   acall bcd2_bcd1          ;正常数据处理程序
asc_sw:   mov a,r4        ;清除有效位前面的 0
     jnz asc_sw_1
     mov r4,#1ch
     mov a,r3
     jnz asc_sw_1
     mov r3,#1ch
asc_sw_1:clr a
asc_out:ret
;------------每字节两位 BCD 码转换为每字节一位 BCD 码子程序
;入口：r3=个位及十位 BCD 码,r4=百位及千位 BCD 码,r6=小数位数
;出口：r1=个位 BCD 码,r2=十位 BCD 码,r3=百位 BCD 码,r4=千位 BCD 码
bcd2_bcd1:mov a,r3       ;个位及十位 BCD 数处理
     anl a,#0fh
     mov r1,a
     mov a,r3
     anl a,#0f0h
     swap a
     mov r2,a
     mov a,r4         ;百位及千位 BCD 数处理
     anl a,#0fh
     mov r3,a
     mov a,r4
     anl a,#0f0h
     swap a
     mov r4,a
     mov a,r6       ;加入小数点信息
     jz bcd2_out
     add a,#ar1
     mov r0,a
     mov a,@r0
     add a,#10h
     mov @r0,a
bcd2_out:ret
;---------------ASCii4_BCD3 转换子程序
;入口：r0=四字节 ASCII 码首地址
;出口：r5,r4,r3 从高至低依次存放 BCD 码（r3,r4 各存 2 位，r5 存 1 位）
ascii4_bcd3:acall ascii4_bin2
     mov dpl,r1
     mov dph,r2
     acall bin_bcd
     ret
;---------------BIN_BCD 转换子程序
;入口：r1=BIN 低字节,r2=BIN 高字节 共 2 字节
;出口：r3=低 2 位 BCD 码，r4=高 2 位 BCD 码,r5=可能的最高位 BCD 码
bin_bcd:PUSH ACC
     PUSH PSW
     mov r3,#0 ;BCD 单元清 0
     mov r4,#0
```

```
     mov r5,#0
     mov r6,#16     ;设置二进制数位数
     clr c
bin_0:    mov a,r1
     rlc a
     mov r1,a
     mov a,r2
     rlc a
     mov r2,a
     mov r0,#ar3
     mov r7,#3      ;设置 BCD 字节数
bin_1:    mov a,@r0
     addc a,@r0
     da a
     mov @r0,a
     inc r0
     djnz r7,bin_1
     djnz r6,bin_0
     POP PSW
     POP ACC
     ret
;---------------ASCII4_BIN2 转换子程序
;入口：R0=ASCII 码首地址（共四字节 ASCII 码）
;出口:r1=BIN 码低字节，R2=BIN 码高字节
ascii4_bin2:PUSH ACC
     acall ascii_bin
     anl a,#0fh
     swap a
     mov r1,a
     inc r0
     acall ascii_bin
     anl a,#0fh
     orl ar1,a
     inc r0
     acall ascii_bin
     anl a,#0fh
     swap a
     mov r2,a
     inc r0
     acall ascii_bin
     anl a,#0fh
     orl ar2,a
     POP ACC
     ret
;---------------ASCII_BIN 转换子程序
;入口：R0=ASCII 地址
;出口:A=BIN 码
ascii_bin:mov a,@r0
     clr c
     subb a,#3ah
     jnc ascii_0
     add a,#7
ascii_0:add a,#3
     ret
     end
```

3、显示、软件看门狗及 20ms 定时器子模块（rand_disp.a51）

```
;--------3--------rand_disp.a51-------------------------
     name   rand_disp
     extrn number(TH0_H,TL0_L,wdt_time)
     PUBLIC I_T0,disp_buf,disp_wc,time_20ms,wdt
  prog SEGMENT CODE          ;定义代码段
  BYTE_BUFF segment data     ;定义字节数据段 1
  byte_buff1 segment idata   ;定义字节数据段 2
  bit_buff segment data bitaddressable          ;定义位数据段

      RSEG BYTE_BUFF
     wdt: ds 1
     time_s_c: ds 1
     time_20ms: ds 1
     DISP_C: ds 11       ;显示位
     disp_buf data disp_c+1 ;个位缓冲区，此个位并非真正的个位，指最右一位，以下类推
     disp_2 data disp_c+2   ;十个位缓冲区
     disp_3 data disp_c+3   ;百位缓冲区
     disp_4 data disp_c+4   ;千位缓冲区
     disp_5 data disp_c+5   ;通道位缓冲区
     disp_6 data disp_c+6   ;地址位缓冲区
     disp_7 data disp_c+7   ;无
     disp_8 data disp_c+8   ;无
     disp_s data disp_c+9   ;闪烁计数器

     rseg bit_buff
     disp_wc: ds 1
     disp_c_1 bit disp_wc.0 ;个位闪烁标志位 1-闪烁
     disp_c_2 bit disp_wc.1 ;十位闪烁标志位
     disp_c_3 bit disp_wc.2 ;百位闪烁标志位
     disp_c_4 bit disp_wc.3 ;千位闪烁标志位
     disp_s_c bit disp_wc.4 ;闪烁控制位   1-闪烁

     RSEG PROG
     using 0
     dsb equ p3.4
     cr_not equ p3.5
     cp equ p3.6
I_T0:     push acc
     push ar0
     push dph
     push dpl
     push psw
     mov th0,#th0_h
     mov tl0,#tl0_l
     mov dptr,#tab
;-----------------------------------------
     djnz disp_s,d_20
     cpl disp_s_c
;-----------------------------------------
d_20:     INC DISP_C
     MOV R0,DISP_C
     CJNE R0,#1,DISP2   ;r0=1 显示个位
     mov c,disp_c_1
     mov A,disp_buf
     mov p1,#0FFH
```

```
     mov p0,#0feh
     ajmp d_10
DISP2:    CJNE R0,#2,DISP3   ;显示十位
     setb p0.0
     mov c,disp_c_2
     mov A,disp_2
     mov p1,#0FFH
     mov p0,#0fdh
     ajmp d_10
DISP3:    CJNE R0,#3,DISP4   ;显示百位
     mov c,disp_c_3
     mov A,disp_3
     mov p1,#0FFH
     mov p0,#0fbh
     ajmp d_10
DISP4:    CJNE R0,#4,DISP5   ;显示千位
     mov c,disp_c_4
     mov A,disp_4
     mov p1,#0FFH
     mov p0,#0f7h
     ajmp d_10
DISP5:    CJNE R0,#5,DISP6   ;显示通道号
     mov A,disp_5
     mov p1,#0FFH
     mov p0,#0efh
     ajmp d_10
DISP6:  CJNE R0,#6,DISP7     ;显示仪表地址
     mov A,disp_6
     mov p1,#0FFH
     mov p0,#0dfh
     ajmp d_10
DISP7:    CJNE R0,#7,DISP8   ;无
     mov A,disp_7
     mov p1,#0FFH
     mov p0,#0bfh
     ajmp d_10
DISP8:    CJNE R0,#8,d_10         ;无
     mov A,disp_8
     mov p1,#0FFH
     mov p0,#7fh
d_10:     acall disp_w
     mov r0,disp_c
     cjne r0,#10,d_12        ;若 disp_c=10 则清 disp_c
     mov disp_c,#0
     inc time_s_c       ;MS,S 计数处理程序
     mov a,time_s_c
     cjne a,#2,d_12
     mov time_s_c,#0
     inc time_20ms
     inc wdt            ;软看门狗处理程序
     mov a,wdt
     cjne a,#wdt_time,d_12
     mov wdt,#0
     mov a,#0
     push acc
     push acc
     reti
;-----------------------
```

```
d_12:    pop psw
     pop dpl
     pop dph
     pop ar0
     pop acc
     RETI
disp_w:  jnc d_0
     jnb disp_s_c,d_0
     mov p1,#0FFH
     ajmp d_1
d_0: movc a,@a+dptr
     cpl a
     mov p1,a
D_1: ret
;-------------------------------------------------------
TAB: DB 3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH
     ;  0   1    2   3   4   5   6   7   8   9
     db 77H,7CH,39H,5EH,79H,71H
     ;   A   B   C   D   E   F

     DB 0BFH,86H,0DBH,0CFH,0E6H,0EDH,0FDH,87H,0FFH,0EFH
     ;   0.   1.  2.   3.   4.   5.   6.   7.   8.  9.
     db 40H,80H,00H,37h,5ch,1ch,58h,74h,0dch,46h
     ;   -   .   空       n   o  u   c    h   a   -1
     ;   1a  1b  1c     1d  1e 1f  20  21  22  23
         END
```

4、命令发送及接收控制子模块（rand_send.a51）

```
;----------4--------rand_send.a51----------------
name send_order
extrn idata (rec_buf)  ;rec_buff接收缓冲区
extrn data (time_20ms)
extrn bit (ri_f)       ;ri_f          正在接收标志
PUBLIC send_order
prog SEGMENT CODE                ;定义代码段
byte_buff segment data           ;定义字节数据段1
byte_buffl segment idata         ;定义字节数据段2
bit_buff segment data bitaddressable ;定义位数据段
     rseg byte_buff
     send_n:ds 1
     order_buf:ds 10
     rseg prog
     dd_c equ 4          ;等待回应20m数
     cw_n equ 7          ;设置允许错误次数
     using 0
;---------------------清命令缓冲区
send_order:
     mov dpl,#8
     mov r0,#order_buf
send_0:  mov @r0,#0
     inc r0
     djnz dpl,send_0
     MOV DPL,#72         ;清接收缓冲区
     MOV R0,#REC_BUF
SEND_A:  MOV @R0,#0
     INC R0
```

```
    DJNZ DPL,SEND_A
;----------------------发送命令控制程序
    mov send_n,#cw_n   ;设置允许错误次数
    mov b,#10
    clr c
    subb a,b
    mov b,#8
    mul ab
    mov dptr,#a_order
    mov r0,#order_buf-1
    mov r1,#10
start_mov:inc r0
    push acc
    movc a,@a+dptr
    mov @r0,a
    inc dptr
    pop acc
    cjne @r0,#0dh,start_mov
;----------------------发送命令子程序
send_start:setb p3.2        ;切换 75176 至发送状态
    mov r0,#order_buf-1
send_10:inc r0
    mov sbuf,@r0
    jnb ti,$      ;等待一帧数据发完
    CLR TI
        cjne @r0,#0dh,send_10   ;检测到结束标志退出
;-----------------------检测接收标志
    mov r0,#30         ;延时 60uS 等待停止位发送完毕
    djnz r0,$
    setb es
    clr p3.2      ;切换 75176 至接收状态
    mov a,#dd_c        ;延时检测仪表有无回应
    add a,time_20ms
send_21:jb ri_f,send_20
    cjne a,time_20ms,send_21
    mov a,#'N'
    ajmp send_23
;------------------------检测接收是否正确
send_20:mov a,#10       ;延时 200MS 等待接收结束
    add a,time_20ms
send_28:jnb ri_f,send_29
    cjne a,time_20ms,send_28
    clr ri_f           ;200MS 仍未结束清接收中标志
send_29:mov r0,#rec_buf+4
    cjne @r0,#'*',send_22
    mov a,#'*'
send_23:djnz send_n,send_start   ;达到允许错误次数退出
    ajmp send_out
send_22:mov r0,#rec_buf+71
    cjne @r0,#0dh,send_23   ;接收缓冲区第 72 字节不为回车符转出错处理
;-----------------------------还原校验码
    mov r0,#rec_buf+70
    mov a,@r0
    clr c
    subb a,#3ah
    jnc SEND_25
    add a,#7
send_25:add a,#3
```

```
     mov @r0,a
     dec r0
     mov a,@r0
     clr c
     subb a,#3ah
     jnc SEND_26
     add a,#7
send_26:ADD A,#3
     swap a
     inc r0
     xchd a,@r0
     dec r0
     xch a,@r0
     inc r0
     mov @r0,#0dh       ;检验码后加回车符
;-----------------------------检验接收缓冲区数据
     mov r0,#rec_buf+1
     mov a,@r0
     inc r0
send_24:xrl a,@r0
     inc r0
     cjne @r0,#0dh,send_24
     cjne a,#0,send_200 ;接收校验未通过转出错处理
send_out:clr es
     ret
send_200:mov a,#1
     ajmp send_23
;-----------------------------------------------------
a_order: db '@0ARD67',0DH   ;读 A 表全部动态数据命令
b_order: db '@0BRD64',0DH   ;读 B 表全部动态数据命令
c_order: db '@0CRD65',0DH   ;读 c 表全部动态数据命令
d_order: db '@0DRD62',0DH   ;读 d 表全部动态数据命令
e_order: db '@0ERD63',0DH   ;读 e 表全部动态数据命令
f_order: db '@0FRD60',0DH   ;读 f 表全部动态数据命令
     END
```

5、串行接收子模块（rand_rec.a51）

```
;-------5----------rand_rec.a51
name rec_si
extrn bit (ri_f)
public rec_si,rec_buf
prog SEGMENT CODE           ;定义代码段
byte_buff segment data          ;定义字节数据段 1
byte_buffl segment idata        ;定义字节数据段 2
bit_buff segment data bitaddressable ;定义位数据段
     rseg byte_buff
     rec_buf_z: ds 1    ;接收区指针
     rec_err: ds 1      ;接收缓冲区溢出计数器
     rseg byte_buffi
     rec_buf: ds 75
     rseg prog
     using 0
rec_si:  PUSH PSW
     push ar0
     push acc
     JNB RI,REC_OUT         ;为发送中断跳出
```

```
     clr ri
     jb ri_f,rec_0
     mov a,sbuf
     cjne a,#'@',rec_out     ;无效命令过滤处理
     setb ri_f               ;置接收中标志
     mov rec_buf_z,#rec_buf  ;接收缓冲区首地址给接收指针
     mov rec_err,#73         ;设置最大接收字节数
     ajmp rec_1
rec_0:    mov a,sbuf
     cjne a,#0dh,rec_1
     clr ri_f
rec_1:    mov r0,rec_buf_z
     mov @r0,a
     inc rec_buf_z
     djnz rec_err,rec_out    ;接收溢出清接收标志
     clr ri_f
rec_out:pop acc
     pop ar0
     POP PSW
     reti
     end
```

6、巡回/定点转换检测子模块（rand_key.a51）

```
;-----------6----rand_key.a51
name key
EXTRN CODE(START)
extrn bit(key_work,key_f)
EXTRN DATA(DISP_buf,disp_wc,time_20ms,wdt)
PUBLIC KEY
prog SEGMENT CODE       ;定义代码段

     RSEG PROG
;-------------------------------------
KEY: mov a,#2
     add a,time_20ms
     cjne a,time_20ms,$
key_1:    mov wdt,#0
     orl 96h,#0e1h
     jnb key_f,key_1
     mov a,#2
     add a,time_20ms
     cjne a,time_20ms,$
     jnb key_f,key_1
     clr ea
     clr ex1
     cpl key_work
;-------------------------------------
key_out:mov dptr,#start
     push dpl
     push dph
     reti
     end
```

7、 rand_js.hex

```
:1000260075820878237600008D582FA75824878406A
:10003600760008D582FA75220775F00AC395F07521
:10004600F008A49000D07822790A08C0E093F6A3BD
:10005600D0E0B60DF5D2B278220886993099FDC265
:1000660099B60DF5781ED8FED2ACC2B27404252F0F
:10007600200907B52FFA744E0193740A252F30090B
:1000860005B52FFAC2097844B62A07742AD522C5BF
:1000960001C97887B60DF67886E6C3943A500224ED
:1000A600072403F618E6C3943A500224072403C42F
:1000B60008D618C608760D7841E6086608B60DFB20
:1000C600B40003C2AC2274010193403041524436 5D
:1000D600370D403042524436340D40304352443698
:1000E600350D403044524436320D40304552443688
:0A00F600330D403046524436300D01
:10010000C0E0C000C083C082C0D0758CFC758A6B13
:100110009001CDD53902B2040530A830B8010CA247
:1001200000E5317590FF7580FE218CB8020ED280FB
:10013000A201E5327590FF7580FD218CB8030CA2F9
:1001400002E5337590FF7580FB218CB8040CA20387
:10015000E5347590FF7580F7218CB8050AE5357593
:1001600090FF7580EF218CB8060AE5367590FF7513
:1001700080DF218CB8070AE5377590FF7580BF21B5
:100180008CB80808E5387590FF75807F31BEA830BF
:10019000B80A20753000052EE52EB40216752E0023
:1001A000052F052DE52DB4640A752D007400C0E0FF
:1001B000C0E032D0D0D082D083D000D0E03250081E
:1001C0003004057590FF21CC93F4F590223F065B37
:1001D0004F666D7D077F6F777C395E7971BF86DBF7
:1001E000CFE6EDFD87FFEF408000375C1C5874DCE4
:0101F00046C8
:1001F1007402252FB52FFD752D004396E130B3F71D
:100201007402252FB52FFD30B3EDC2AFC2AAB208DB
:08021100900257C082C0833245
:03000000020219E0
:0100030032CA
:03000B00020100EF
:040013000201F132C3
:01001B0032B2
:0300230002039E37
:100219004396E17580FF78207400F608B880FBD218
:100229000875B0FF75A0FF75200075B81275985054
:1002390075892175 8CFC758A6B758DFD758BFD75BE
:10024900A882758700D28ED28C7179439602C2AF8B
:10025900758107D2AFC209C2B2300857D2AA753D1B
:100269000A753C005320F0752D00439602E53D12B6
:1002790000267033853D36E53C1203D3701F853C5B
:1002890035717075 2D00439602744B252FB52FFDDE
:10029900752D00439602744B252FB52FFD053CE5BE
:1002A9003CB408C0753C00053DE53DB410B6753D4C
:1002B9000A416DD2AA432001752D0043960220B44C
:1002C900457402252FB52FFD20B43B752D004396AB
:1002D9000230B4F7E53CB4070F753C00053DE53D38
:1002E900B41007753D0A41F3053CE53D120026703F
:1002F900EB853D36E53C1203D370D9853C357170E9
:100309007464252F41C120B5457402252FB52FFDF1
:1003190020B5A5752D0043960230B5F7E53CB4002C
:100329000F753C07153DE53DB40907753D0F613B68
:10033900153CE53D12002670EB853D36E53C120380
:10034900D370D9853C3571707464252F41C1B52F9F
:1003590014E53D120026700DE53C1203D3700671B9
:10036900707464252F41C189318A328B338C3422D0
:10037900791E7A227B217C1F753521753620717093
:10038900744B252FB52FFD752D00439602744B250F
:050399002FB52FFD222D
:10039E00C0D0C000C0E0309825C29820090FE59962
:1003AE00B4401BD209753E40753F4961C2E599B410
:1003BE000D02C209A83EF6053ED53F02C209D0E0A5
:0503CE00D000D0D03288
:1003D30075F008A42405784028F5F0F8915AE5F063
:1003E3002407F891A8FEED6043E582F4C304F58287
:1003F300E583F43400F5837011E58214700C790FF2
:100403007A0F7B0F7C0F740F813CA982AA839163BF
:10041300E5F02407F891A8FE913DEC1470047C23C9
:1004230081317C237B097A197909813B913DEC70F9
:10043300077C1CEB70027B1CE422EB540FF9EB549A
:10044300F0C4FAEC540FFBEC54F0C4FCEE60072448
:1004530001F8E62410F622918889828A8391632227
:10046300C0E0C0D07B007C007D007E10C3E933F97F
:10047300EA33FA78037F03E636D4F608DFF9DEEDD4
:10048300D0D0D0E022C0E091A8540FC4F90891A8BD
:10049300540F42010891A8540FC4FA0891A8540FAD
:1004A3004202D0E022E6C3943A50022407240322F6
:00000001f
```