第 24 卷 第 1 期 华 侨 大 学 学 报 (自 然 科 学 版) Vol. 24 No. 1
2003 年 1 月 Journal of Huaqiao University (Natural Science) Jan. 2003

文章编号 1000-5013(2003) 01-0098-07

C 语言实现单片机的数据处理及应用

龚 冬 梅

(华侨大学信息科学与工程学院, 福建 泉州 362011)

摘要 介绍一种用 C 语言实现单片机数据处理的方法, 它能重点实现对不同型号热电偶分度表的自动查表、热电偶温度测量的非线性修正, 以及用热敏电阻换算法, 进行冷端温度自动补偿计算. 文中还论述数组指针在数据表格中、插值计算在非线性修正中的应用. 实践表明, 用软件数据处理代替传统的硬件测量系统, 具有硬件结构简单、使用灵活、测量精度高等优点.

关键词 C51, 数据处理, 热电偶, 温度测量

中图分类号 TP 368.2 ; TN 377 文献标识码 A

MCS-51 系列单片机被广泛应用于仪器仪表、工业控制、智能终端等诸多领域. 随着软件技术的迅猛发展, 要求单片机系统应具有一定的数据处理能力. 常用的汇编语言难以胜任大数据的管理和复杂的数学运算, 而且程序可读性和可移植性也比较差. 目前 C 语言基本上克服了上述缺陷, 使程序大大简化[1], 同时在不增加硬件就能改变或适应不同的对象控制和测量. 工业用的绝大多数热电偶的热电动势与对应的温度之间的关系为非线性, 且难以找到一个确切的解析式来表达温度 t 与热电动势 E 之间的函数关系 $E=f(t)$. 所以, 工程上一般都是以热电偶分度表的形式, 给出它们之间的对应关系. 在自动检测系统中人们采用许多非线性修正方法, 特别是采用计算机数据处理的方法, 其应用尤为广泛[2~3]. 因此, 本文以热电偶为例, 介绍一种用 C 语言实现单片机数据处理及应用于温度自动测量的方法.

1 单片机数据表格的访存

1.1 单片机 C 语言数据表格的特点

数据表格的访存是数据处理的基本内容. 汇编语言对数据表的访问, 是通过 MOVC A, @A+ DPTR 和 MOVC A, @A+ PC 查表指令来实现的. 它可直接查找地址范围为 256 个单元字节, 而一个高精度数据往往要使用 2~4 个地址单元来存储. 因此, 用汇编语言进行数据表格的管理既不方便, 又不直观. C 语言访问数据表格最简捷的方式是采用指针变量或数组变量, 将数据表格定义为数组. 一个表格可定义为一个一维数组, 也可按行列定义为二维数组. 当有多个表格时, 可将数组维数加 1, 或另外定义一个数组. 单片机数据表格结构不同于常规的

收稿日期 2002-08-29 作者简介 龚冬梅(1968-), 女, 实验师

数据库结构, 因为单片机的数据存储能力和运算速度远远低于目前的 PC 机. 所以, 数据表格应考虑结构紧凑, 无空间浪费, 并且在数据查表时应考虑数据查询速度. 数组形式的数据表格, 其元素所占的字节数 d 是固定的(对整形 d= 2, 对实型 d= 4, 对字符型 d= 1), 适合于存放诸如热电偶分度表等表格. 如果用相同的数组存放不同热电偶的热电动势表, 则会造成很大的空间浪费. 因此, 应对不同热电偶分度表分别定义不同数组.

1. 2　单片机 **C** 语言数据表格的引用

引用数组元素可以用下标法, 也可以用指针法. 即通过指向数组元素的指针找到所需的元素, 使用指针法能使目标程序质量高(占内存少, 运行速度快). 如果指针变量 p 指向数组中的某一个元素, p + 1 则指向同一数组中的下一个元素. 例如, 数组元素是实型, 每个数组元素占 4 B, 则 p + 1 意味着 p 的值(地址) 加 4 B. 在访问数组时, 指针变量与指针常量在表现形式和含义上有相同之处, 但操作上又不尽相同. 与指针常量 (地址) 相对应的指针变量也有它的"级别"[4], 指针变量的级别是由其定义方式决定. "一级"指针变量指向数组元素, 即一级指针"按元素移动", 如 int(* p). "二级"指针变量指向数组行, 设有说明 int(* p)[n], 则 p 是一个"二级"指针变量. 它指向一个包含 n 个元素的一维数组, 其值只能是二级地址 (行地址). 此时, p+ 1 指向"下一行", 即二级指针"按行移动". 如果一个数据表的数据(即数组中的元素) 很多, 先用"二级"指针进行粗略搜索, 然后再用"一级"指针搜索具体元素. 这样, 可以提高查表速度.

2　热电偶温度测量的应用程序

2. 1　热电偶分度表定义为 **C** 语言数据表格

工业用热电偶的测温精度可达 0. 1 ℃. 如果按 0. 1 ℃间隔存储热电偶热电动势值, 该数据表会很庞大. 在工程应用中, 通常按 10 ℃间隔存储热电偶热电动势值, 其它温度的热电动势值按该温度段的线性关系计算. 因此单片机存储的热电偶热电动势表就是按 10 ℃步进的顺序表格. 为了实现无空间浪费, 这里采用不同热电偶分别定义和存放表格数据的方法, 并按二维数组存放, 每行 10 个元素. 例如, 铂铑$_{10}$-铂(S) 热电偶的测温范围一般为 0～1 600 ℃, 其分度表[5]可定义为一个 16×10 数组; 镍铬-镍硅(K) 热电偶的测温范围为- 200～1 200 ℃, 其分度表可定义为一个 14×10 数组. 数组模块为

float xdata ELE - S[16][10] = {{0. 000, 0. 055, 0. 113, 0. 173, 0. 235, 0. 299, 0. 365, 0. 433, 0. 502, 0. 573}, {0. 646, 0. 720, 0. 795, 0. 872, 0. 950, 1. 029, 1. 110, 1. 191, 1. 273, 1. 357}, …};

float xdataELE-K[14][10] = {{- 5. 891, - 5. 730, - 5. 550, - 5. 354, - 5. 141, - 4. 913, - 4. 669, - 4. 411, - 4. 138, - 3. 852}, {- 3. 554, - 3. 243, - 2. 920, - 2. 587, - 2. 243, - 1. 889, - 1. 527, - 1. 156, - 0. 778, - 0. 392}, {0. 000, 0. 397, 0. 798, 1. 203, 1. 612, 2. 023, 2. 436, 2. 851, 3. 267, 3. 682}, …};

2. 2　已知热电动势查找相应温度的程序模块

已知热电动势 E, 根据不同数据表格查找对应的行 i 和列 j, 再计算出相应的温度. 例如在 ELE - S 数组中对应的温度是(100× i+ 10× j) ℃, 在 ELE - K 数组中对应的温度是(- 200+

$100\times i+10\times j$) ℃. 程序模块为

```
/*  已知热电动势 E, 求 i, j, k 程序 */
# include< reg52.h>
# include< stdio.h>
extern float xdata RT[][10];        /* 声明热敏电阻电压温度数据表为外部数组*/
extern float xdata ELE-S[][10];     /* 声明铂铑-铂热电动势表为外部数组*/
extern float xdata ELE-K[][10];     /* 声明镍铬-镍硅热电动势表为外部数组*/
……
int i, j, k;
Tsearch()
{int s;
 int (*p)[10];                                  /* 定义二级指针 p[10]*/
 switch(XH)
 case 'R': p=RT;       break;                   /* 'R'为热敏电阻 */
 case 'S': p=ELE-S;    break;                   /* 'S'为S型热电偶 */
 case 'K': p=ELE-K;    break;                   /* 'K'为K型热电偶 */
   ……
   default: p=ELE-S;}
 for(s=0;s<20;s++)
    {if(E<*p[s]),                               /* 数组第 s 行第一个元素*/
    i=s-1,s=20;}
 for (s=0;s<10;s++)
    {if(E<*(p[i]+s),                            /* 数组第 i 行第 s 个元素*/
    j=s,s=10;}
    k=10*i+j;  while(1);}
```

2.3 已知温度查找相应热电动势的程序模块

已知温度 T, 根据不同数据表格计算出对应温度 T 的行 i 和列 j, 再查表得相应的热电偶热电动势 E. 例如, 对于 ELE-S 计算公式为 $i=(T/100)$, $j=(T-100i)/10$; 对于 ELE-K 计算公式为: $i=(T/100)+2$, $j=(T+200-100i)/10$. 程序模块为

```
/*  已知 i, j 求热电动势 E 程序 */
……
extern float xdata RT[][10];
extern float xdata ELE-S[][10];
extern float xdata ELE-K[][10];
……
Esearch()
{switch(XH)
 case 'S': e=ELE-S[i][j];   break;
```

```
    case 'K': ex= ELE-K[i][j]; break;
    ……
      default: ex= ELE-S[i][j];}…
  while(1);}
```

3　插值计算非线性修正与冷端补偿计算

3.1　插值计算方法

在自动检测系统中, 传感器的输出非线性修正可以用硬件实现, 也可以用软件实现. 由于 C 语言相对于汇编语言的最大优点就是数学计算程序简单, 并有丰富的库函数可以调用, 所以用软件实现是最理想的选择. 为了得到高的测量精度, 计算热电偶实际温度测量值时要用其热电动势分度表数据进行插值计算. 插值计算方法通常采用线性插值和样条插值. 这里采用拉格朗日插值方法, 其公式为

$$f(x)=\frac{(x-x_1)(x-x_2)\cdots(x-x_n)}{(x_0-x_1)(x_0-x_2)\cdots(x_0-x_n)}y_0+\frac{(x-x_0)(x-x_2)\cdots(x-x_n)}{(x_1-x_0)(x_1-x_2)\cdots(x_1-x_n)}y_1+\cdots+\frac{(x-x_0)(x-x_1)\cdots(x-x_{n-1})}{(x_n-x_0)(x_n-x_1)\cdots(x_n-x_{n-1})}y_n+R_n(x). \tag{1}$$

在满足精度要求前提下, 为了减少运算量, 我们选用分段插值. 每一段选 3 个插值点(热电动势值) E_{k-1}, E_k, E_{k+1}, 对应温度值为 T_{k-1}, T_k, T_{k+1}, 并使 $E_{k-1}<E<E_{k+1}$, 其中 E 为测量热电动势值. 每一段的拉格朗日插值公式可近似表示为

$$T(E)\approx\frac{(E-E_x)(E-E_{k+1})}{(E_{k-1}-E_k)(E_{k-1}-E_{k+1})}T_{k-1}+\frac{(E-E_{k-1})(E-E_{k+1})}{(E_k-E_{k-1})(E_k-E_{k+1})}T_k+\frac{(E-E_{k-1})(E-E_k)}{(E_{k+1}-E_{k-1})(E_{k+1}-E_k)}T_{k+1}. \tag{2}$$

由于 T_k～E_k 关系已存于表格中, 对任何一个热电动势测量值 E, 都可以找到一个相近的 E_k 值及对应的温度 T_k. 然后根据式(2), 计算精确的温度值 $T(E)$.

3.2　插值计算程序

下面是插值计算相关程序, 其中数组 $X[3]$, $Y[3]$ 对应下标为 $k-1$, k, $k+1$ 的温度或热电偶热电动势值(见式(2)), 定义 Calculate() 为插值函数. 使用时, 先给 $X[3]$, $Y[3]$ 赋值. 为了保证精度, 采用浮点运算, 然后将 Calculate() 按外部函数调用. 有

```
……
float X[3]={E(k-1),E(k),E(k+1)};                    /* 给 X[ ] 赋值* /
float Y[3]={T(k-1),T(k),T(k+1)};…                   /* 给 Y[ ] 赋值* /
void calculate(void)
{float DX0=X-X[0], DX1=X-X[1], DX2=X-X[2];
float X0=X[1]-X[0], X1=X[2]-X[1], X2=X[0]-X[2];
float A=(DX1*DX2)/((-X0)*X2);
```

```
float B=(DX0*DX2)/(X0*(-X1));
float C=(DX0*DX1)/((-X2)*X1);
Y=A*Y[0]+B*Y[1]+C*Y[2];…
while(1);}
```

3.3 冷端补偿计算

由于热电偶的热电动势分度表是在冷端温度为0 ℃的情况下得到的. 因此, 冷端温度不等于0 ℃时, 就需要对测量值进行补偿. 如冷端温度为 t_0(t_0> 0 ℃), 则可利用下式进行修正为

$$E(T,0\ ℃) = E(T,t_0) + E(t_0,0\ ℃), \tag{3}$$

式中 $E(T,0\ ℃)$ 为标准热电动势值, $E(T,t_0)$ 为实际测量热电动势值, $E(t_0,0\ ℃)$ 为冷端补偿热电动势值. 常用的冷端补偿有冰浴法、补偿电桥法等. 硬件补偿法. 结构复杂, 并且要经常修正. 软件补偿可采用热敏电阻换算法. 也即将置于冷端温度 t_0 环境下的热敏电阻的电阻测量值 $R(t_0)$, 通过软件换算为冷端补偿热电动势值 $E(t_0,0\ ℃)$. 热敏电阻测量电路原理, 如图1所示. 热敏电阻的电阻值 R_t 可以通过其两端电压值 $V(t_0)$ 输入到单片机, $V(t_0)=IR_t$. 当 I 由恒流源提供时(I 为常数), 在较窄温度范围内, R_t 与 $V(t_0)$ 成线性关系. 对测量到的 R_t, 已转换为 $V(t_0)$. 根据热敏电阻的电阻-温度特性曲线(同样制成表格存放在单片机数据表格中), 用插值计算法求出相应的冷端温度 t_0. 再由热电偶热电动势表通过插值计算, 求出相应于冷端温度 t_0 的冷端补偿热电动势值 $E(t_0,0\ ℃)$. 虽然进行了多次插值计算, 但是由于C语言的模块化设计和函数功能, 实际编程时只须调用插值函数即可.

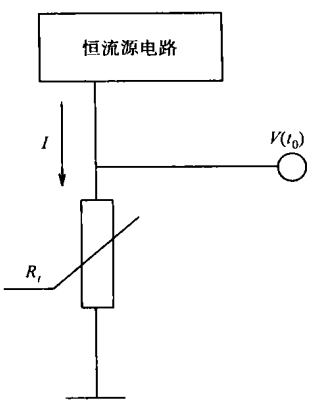

图1 热敏电阻测量电路图

4 热电偶温度测量数据处理程序流程图及数据采集硬件框图

热电偶温度测量数据采集硬件框图, 如图2所示. 数据处理程序流程如图3所示. 程序的

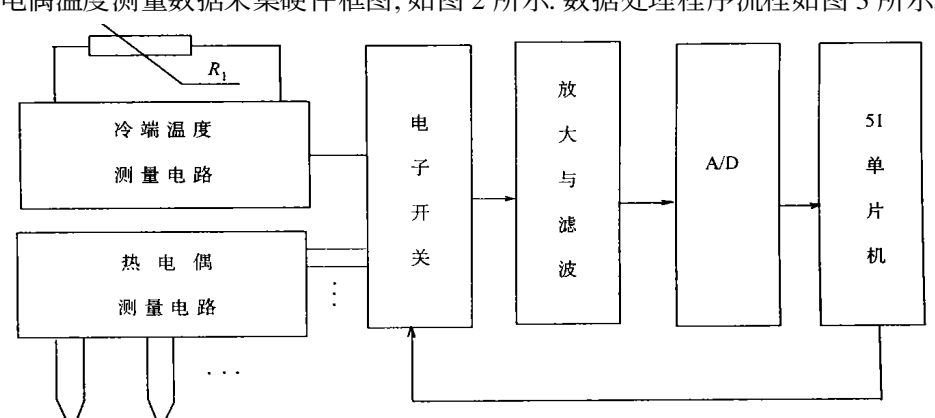

图2 热电偶温度测量数据采集硬件框图

第1部分是冷端补偿. 它是先用查表法确定被测温度的粗值 t_{tk}, 然后用插值计算得到冷端精确

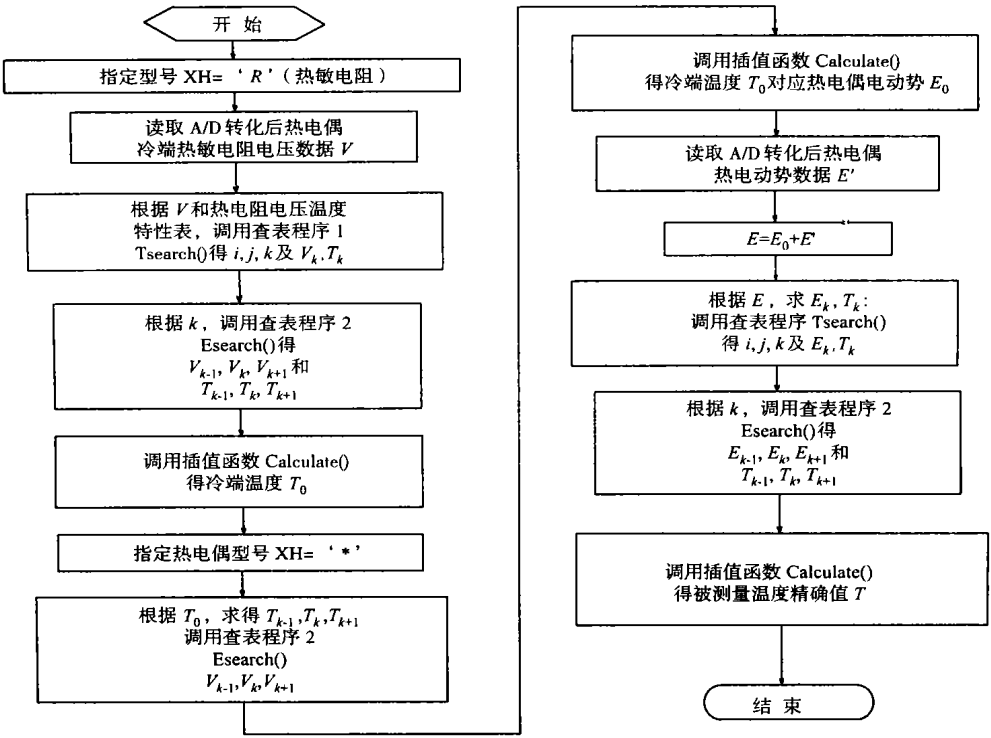

图 3　温度测量数据处理程序流程图

温度值 t_0, 再将 t_0 换算成热电偶热电动势值 $E(t_0, 0\ ℃)$. 程序第 2 部分是用查表法先确定被测温度的粗值 T_k, 尔后用插值计算得到精确温度值 T.

5　数据处理精度比较

为了比较插值计算非线性修正前后温度测量数据的精度, 采用模拟热电偶电势输入方法. 通过数据采集和数据处理, 以铂铑$_{10}$–铂(S 型) 热电偶为例, 测得一组数据如表 1 所示. 对应一个实际温度值 t, 从铂铑$_{10}$–铂(S 型) 分度表查得相应热电动势值 $E(t)$. 该电动势由高稳定度可调电压源产生, 并从热电偶输入端输入, 经 A/D 转换后由单片机进行数据采集和处理. 表中修正前温度测量值为采用 10 ℃间隔直线拟合法测量值, 修正后温度测量值为采用 10 ℃间隔拉格朗日插值计算修正测量值. 可以看出, 修正前测量值平均误差(均方差) 比修正后测量值平均误差(均方差) 大. 这说明采用插值计算非线性修正可以提高热电偶温度测量精度.

表 1　铂铑–铂(S) 热电偶非线性修正前后测量精度数据对比表

t/(℃)	$E(t)$/mV	修正前温度测量值/(℃)	修正前误差温度/(℃)	修正后温度测量值/(℃)	修正后误差温度/(℃)
65.0	0.399	65.1	+ 0.1	65.0	0
105.0	0.683	105.2	+ 0.2	105.1	+ 0.1
164.0	1.142	163.9	− 0.1	164.0	0
234.0	1.733	233.8	− 0.2	234.0	0

续表

t/(℃)	$E(t)$/mV	修正前温度测量值/(℃)	修正前误差温度/(℃)	修正后温度测量值/(℃)	修正后误差温度/(℃)
336.0	2.655	336.3	+0.3	336.1	+0.1
475.0	3.987	475.0	0	475.0	0
566.0	4.893	565.8	−0.2	565.9	−0.1
703.0	6.307	703.4	+0.4	703.1	+0.1
857.0	7.970	856.8	−0.2	857.0	0
996.0	9.541	996.1	+0.1	996.0	0
1 084.0	10.567	1 083.7	−0.3	1 083.9	−0.1
1 137.0	11.196	1 136.7	−0.3	1 136.8	−0.2
	平均误差(均方差)		0.23		0.08

6 结束语

用C51编程可以使单片机具有较强大的数据处理能力.本文所阐述的单片机数据表格处理和插值计算非线性修正,它是一个应用于热电偶温度测量的成功例子.而用软件数据处理代替传统的硬件测量系统,具有硬件结构简单、使用灵活、测量精度高等优点.此法也适用于非线性传感器的线性化处理.

参 考 文 献

1 赖麒文.8051单片机C语言开发环境实务与设计[M].北京:科学出版社,2002.290～450
2 王 霄.实现热电偶电势非线性补偿的软件方法[J].耐火材料,1998,32(2):36～38
3 陈 羿,周东祥.热电偶热电势——温度特性的线性化处理[J].仪表技术与传感器,1999,3(4):31～38
4 刘 卫.C语言的数组与指针[J].西南工学院学报,2000,15(3):21～24
5 林宗虎.工程测量技术手册[M].北京:化学工业出版社,1997.87～110

C-Language Implemented Data Processing by One-Chip Microprocessor and Its Application

Gong Dongmei
(College of Info. Sci. & Eng., Huaqiao Univ., 362011, Quanzhou, China)

Abstract Measuring temperature with thermocouple, a method of C-language implemented data processing by one-chip microprocessor is presented, laying stress on application of array pointer to data list and application of interpolation calculation to nonlinear correction. By this method, automatic look up different types of thermocouple scale lists can be realized; nonlinear correction themocouple measuring the temperature and autocompensatedly calculating the temperature at cold-junction by thermistor conversion can also be realized. As indicated by practice, the use of software data processing as a substitute for conventional hardware measuring system is characterized by simple in hardware structure, flexible in service, and high in measuring precision.

Keywords C51, data processing, thermocouple, temperature measurement