

第 13 章 LabVIEW 串口通信程序设计

以 PC 作为上位机，以调制解调器 (Modem)、串行打印机、各种监控模块、PLC、摄像头云台、数控机床、单片机及智能设备等作为下位机广泛应用于测控领域。本章举几个典型实例，详细介绍利用 LabVIEW 实现 PC 与各种下位机设备串口通信的程序设计方法。

13.1 PC 与 PC 串口通信

当两台串口设备通信距离较近时，可以直接连接，最简单的情况，在通信中只需三根线（发送线、接收线、信号地线）便可实现全双工异步串行通信。

本设计通过两台 PC 串口三线连接，介绍了串口通信的基本编程方法。

13.1.1 PC 与 PC 串口通信硬件线路

当两台 RS-232 串口设备通信距离较近时 (<15m)，可以用电缆线直接将两台设备的 RS-232 端口连接；若通信距离较远 (>15m) 时，需附加调制解调器 (Modem)。

在 RS-232 的应用中，很少严格按照 RS-232 标准。其主要原因是因为许多定义的信号在大多数的应用中并没有用上。在许多应用中，例如 Modem，只用了 9 个信号（两条数据线、6 条控制线、一条地线）；在其他一些应用中，可能只需要 5 个信号（两条数据线、两条握手线、一条地线）；还有一些应用，可能只需要数据线，而不需要握手线，即只需要 3 个信号线。因为在控制领域，在近距离通信时常采用 RS-232，所以这里只对近距离通信的线路连接进行讨论。

当通信距离较近时，通信双方不需要 Modem，可以直接连接，这种情况下，只需使用少数几根信号线。最简单的情况，在通信中根本不需要 RS-232C 的控制联络信号，只需三根线（发送线、接收线、信号地线）便可实现全双工异步串行通信。

在实际使用中常使用串口通信线将两个串口设备连接起来。串口线的制作方法非常简单：准备两个 9 针的串口接线端子（因为计算机上的串口为公头，因此连接线为母头），准备 3 根导线（最好采用 3 芯屏蔽线），按图 13-1 所示将导线焊接到接线端子上。

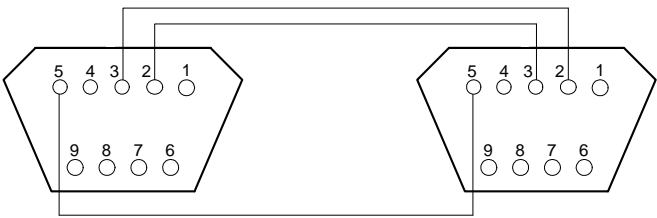


图 13-1 串口通信线的制作

图 13-2 所示中的 2 号接收脚与 3 号发送脚交叉连接是因为在直连方式时,把通信双方都当作数据终端设备看待,双方都可发也可收。在这种方式下,通信双方的任何一方,只要请求发送 RTS 有效和数据终端准备好 DTR 有效就能开始发送和接收。



图 13-2 PC 与 PC 串口通信线路

在计算机通电前,按图 13-2 所示将两台 PC 的 COM1 口用串口线连接起来。



连接串口线时,计算机严禁通电,否则极易烧毁串口。

13.1.2 设计任务

利用 LabVIEW 编写程序实现 PC 与 PC 串口通信。

任务要求如下。

两台计算机互发字符并自动接收,如一台计算机输入字符串“收到信息请回字符 abc123”,单击“发送字符”命令,另一台计算机若收到,就输入字符串“收到, abc123”,单击“发送字符”命令,信息返回到第一组的计算机。

实际上就是编写一个简单的双机聊天程序。

13.1.3 任务实现

1. 建立新 VI 程序

启动 NI LabVIEW 程序,选择新建(New)选项中的 VI 项,建立一个新 VI 程序。

2. 程序前面板设计

☞ 在前面板设计区空白处单击鼠标右键,显示控件选板(Controls)。

(1) 添加一个字符串输入控件: 控件(Controls)→新式(Modern)→字符串与路径(String & Path)→字符串输入控件(String Control),将标签改为“发送区:”。

(2) 添加一个字符串显示控件: 控件(Controls)→新式(Modern)→字符串与路径(String & Path)→字符串显示控件(String Indicator),将标签改为“接收区:”。

(3) 添加一个串口资源检测控件: 控件(Controls)→新式(Modern)→I/O→VISA 资源名称(VISA resource name);单击控件箭头,选择串口号,如 COM1 或 ASRL1:。

(4) 添加一个确定(OK)按钮控件: 控件(Controls)→新式(Modern)→布尔(Boolean)→确定按钮(OK Button),将标题改为“发送字符”。

(5) 添加一个停止(Stop)按钮控件: 控件(Controls)→新式(Modern)→布尔(Boolean)→

停止按钮 (Stop Butoon)，将标题改为“关闭程序”。

设计的程序前面板，如图 13-3 所示。

3. 框图程序设计——添加函数

④ 进入框图程序设计界面，在设计区的空白处单击鼠标右键，显示函数选板（Functions）。添加的所有函数及其布置如图 13-4 所示。

详细步骤介绍如下。

(1) 添加一个配置串口函数：编程 (Programming) → 图 13-3 程序前面板
仪器 I/O (Instrument I/O) → 串口 (Serial) → VISA 配置串口 (VISA Configure Serial Port)。

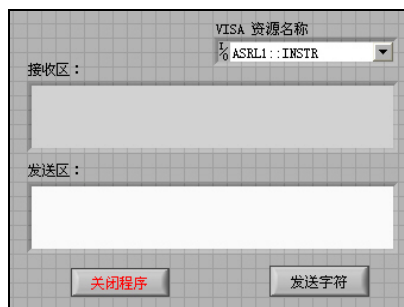


图 13-3 程序前面板

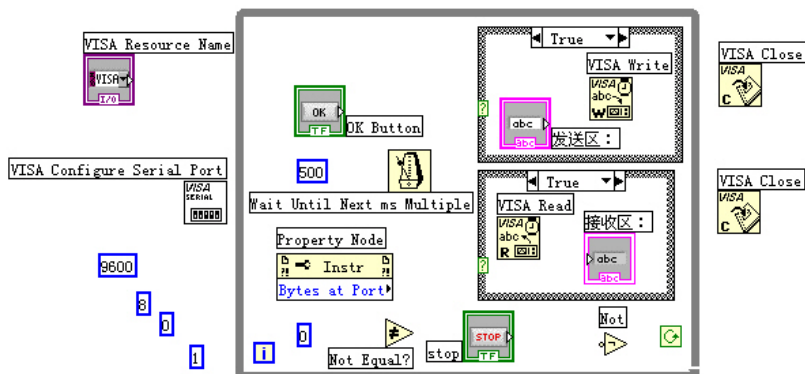


图 13-4 框图程序函数添加与布置

(2) 添加 4 个数值常量：编程 (Programming) → 数值 (Numeric) → 数值常量 (数值常量 (Numeric Constant)，值分别为 9600 (波特率)、8 (数据位)、0 (校验位，无)、1 (停止位)。

(3) 添加两个关闭串口函数：编程（Programming）→仪器 I/O（Instrument I/O）→串口（Serial）→VISA 关闭（VISA Close）。

(4) 添加一个循环结构: 编程 (Programming) → 结构 (Structures) → While 循环 (While Loop)。添加理由: 随时监测串口接收缓冲区的数据。

以下添加的函数或结构放置在 While 循环结构框架中。

(5) 添加一个时钟函数：编程 (Programming) → 定时 (Timing) → 等待下一个整数倍毫秒 (Wait Until Next ms Multiple)。添加理由：以一定的周期监测串口接收缓冲区的数据。

(6) 添加一个数值常量: 编程 (Programming) → 数值 (Numeric) → 数值常量 (Numeric Constant), 将值改为 500 (时钟频率值)。

(7) 添加一个 VISA 串口字节数函数：编程 (Programming) → 仪器 I/O (Instrument I/O) → 串口 (Serial) → VISA 串口字节数 (VISA Bytes at Serial Port)，标签为 “Property Node”。

(8) 添加一个数值常量：编程 (Programming) → 数值 (Numeric) → 数值常量 (Numeric Constant)，将值为 0 (比较值)。

(9) 添加一个比较函数：编程 (Programming) → 比较 (Comparison) → 不等于? (Not Equal?)。添加理由：只有当串口接收缓冲区的数据个数不等于 0 时，才将数据读入到接收区。

(10) 添加一个布尔函数：编程 (Programming) \rightarrow 布尔 (Boolean) \rightarrow 非 (Not) 函数。

添加理由：当关闭程序时，将关闭按钮真（True）变为假（False），退出循环。如果将循

环结构的条件端子  设置为“真时停止 (Stop if True)” ，则不需要添加非 (Not) 函数。

(11) 添加两个条件结构：编程 (Programming) → 结构 (Structures) → 条件结构 (Case Structure)。添加理由：发送字符时，需要单击按钮“发送字符”，因此需要判断是否单击了发送按钮；接收数据时，需要判断串口接收缓冲区的数据个数是否不为 0。

(12) 添加一个串口写入函数：编程 (Programming) → 仪器 I/O (Instrument I/O) → 串口 (Serial) → VISA 写入 (VISA Write)，并拖入条件结构 (上) 的真 (True) 选项框架中。

(13) 添加一个串口读取函数：编程 (Programming) → 仪器 I/O (Instrument I/O) → 串口 (Serial) → VISA 读取 (VISA Read)，并拖入条件结构 (下) 的真 (True) 选项框架中。

(14) 将字符输入控件图标 (标签为“发送区:”) 拖入条件结构 (上) 的真 (True) 选项框架中，将字符显示控件图标 (标签为“接收区:”) 拖入条件结构 (下) 的真 (True) 选项框架中。

(15) 分别将确定 (OK) 按钮控件图标 (标签为“确定按钮 (OK Button)”)、停止 (Stop) 按钮控件图标 (标签为“停止按钮 (Stop Button)”) 拖入循环结构框架中。

4. 框图程序设计——连线

使用连线工具，将所有函数连接起来，如图 13-5 所示。

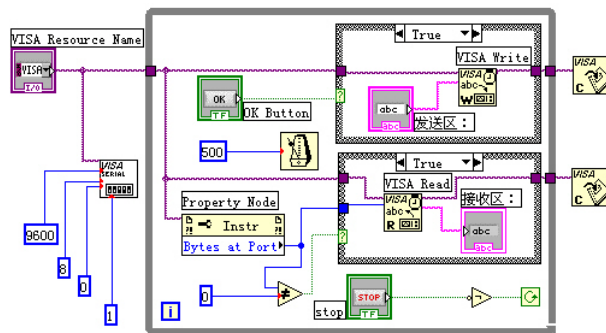


图 13-5 框图程序连线

(1) 将 VISA 资源名称 (VISA resource name) 函数的输出端口分别与串口配置 (VISA Configure Serial Port) 函数、串口字节数 (VISA Bytes at Serial Port) 函数、串口写入 (VISA Write) 函数、串口读取 (VISA Read) 函数的输入端口 VISA 资源名称 (VISA resource name) 相连。

(2) 将数值常量 9600、8、0、1 分别与串口配置 (VISA Configure Serial Port) 函数的输入端口波特率 (baud rate)、数据比特 (data bits)、奇偶 (parity)、停止位 (stop bits) 相连。

(3) 将数值常量 (值为 500) 与等待下一个整数倍毫秒 (Wait Until Next ms Multiple) 函数的输入端口毫秒倍数 (millisecond multiple) 相连。

(4) 将确定按钮图标“OK Button”与条件结构 (上) 的选择端子? 相连。

(5) 将串口字节数 (VISA Bytes at Serial Port) 函数的输出端口 Number of bytes at Serial port 与不等于? (Not Equal?) 函数的输入端口 x 相连。

将串口字节数 (VISA Bytes at Serial Port) 函数的输出端口 Number of bytes at Serial port 与串口读取 (VISA Read) 函数的输入端口字节总数 (byte count) 相连。

(6) 将数值常量 (值为 0) 与不等于? (Not Equal?) 函数的输入端口 y 相连。


(7) 将不等于? (Not Equal?) 函数的输出端口 $x \neq y$ 与条件结构 (下) 的选择端子? 相连。

(8) 在条件结构 (上) 中将字符输入控件图标 (标签为“发送区:”) 与串口写入 (VISA

Write) 函数的输入端口写入缓冲区 (write buffer) 相连。

(9) 在条件结构 (下) 中将串口读取 (VISA Read) 函数的输出端口读取缓冲区 (read buffer) 与字符显示控件图标 (标签为 “接收区:”) 相连。

(10) 将停止按钮 (Stop Button) 函数与非 (Not) 函数的输入端口 x 相连。

(11) 将非 (Not) 函数的输出端口 .not. x? 与循环结构的条件端子  相连。

(12) 在条件结构 (上) 中将串口写入 (VISA Write) 函数的输出端口 VISA 资源名称输出 (VISA resource name out) 与串口关闭 (VISA Close) 函数 (上) 的输入端口 VISA 资源名称 (VISA resource name) 相连。

(13) 在条件结构 (下) 中将串口读取 (VISA Read) 函数的输出端口 VISA 资源名称输出与关闭串口函数 VISA Close (下) 的输入端口 VISA 资源名称相连。

(14) 进入两个条件结构的假 (False) 选项, 将 VISA 资源名称函数的输出端口分别与串口关闭 (VISA Close) 函数 (上、下) 的输入端口 VISA 资源名称相连, 如图 13-6 所示。

5. 运行程序

进入程序前面板, 保存设计好的 VI 程序。单击快捷工具栏 “运行 (Run)” 按钮, 运行程序。



两台计算机同时运行本程序。

在一台计算机程序窗体中发送字符区输入要发送的字符, 比如 “收到信息请回字符 abc123”, 单击 “发送字符” 按钮, 发送区的字符串通过 COM1 口发送出去。

如果联网通信的另一台计算机程序收到字符, 则返回字符串, 如 “收到, abc123”; 如果通信正常该字符串将显示在接收区中。

程序运行界面如图 13-7 所示。

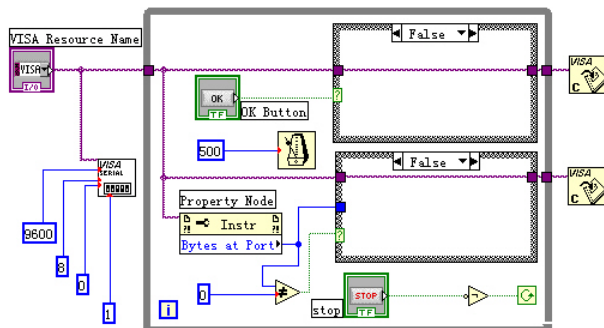


图 13-6 框图程序连线

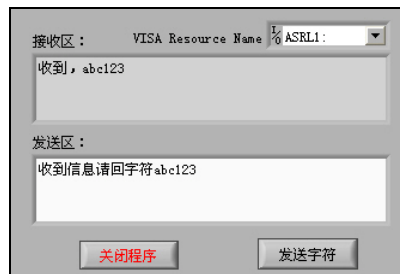


图 13-7 程序运行界面

6. 单 PC 双串口互通程序

如果只有一台计算机且具有两个串口, 那么可以通过串口线将两个串口直接连接起来, 如图 13-8 所示, 编写程序实现双串口互通信。

图 13-9 是单 PC 双串口互通信程序的前面板。

图 13-10 是单 PC 双串口互通信程序的后面板。

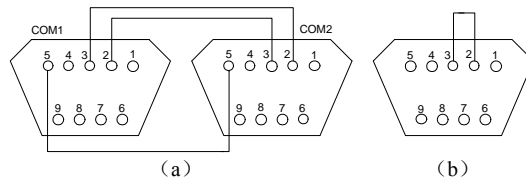


图 13-8 双串口直接连接

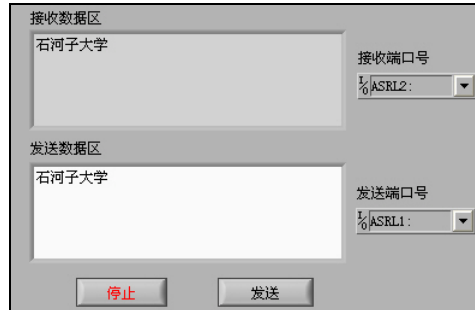


图 13-9 单 PC 双串口互通信程序的前面板

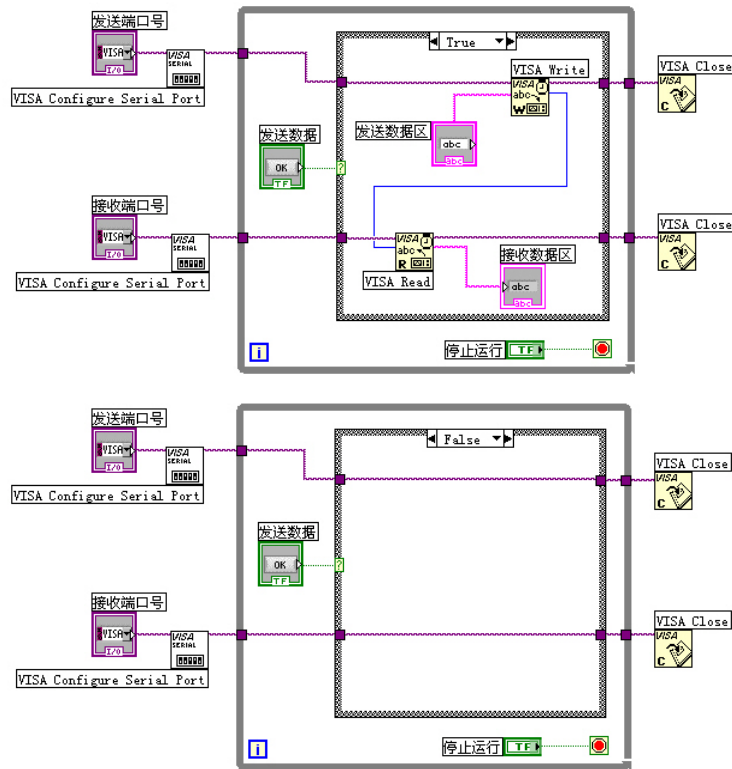


图 13-10 单 PC 双串口互通信程序的后面板

13.2 PC 与单片机串口通信程序设计

目前，在许多单片机应用系统中，上、下位机分工明确，作为下位机核心器件的单片机

往往只负责数据的采集和通信，而上位机通常以基于图形界面的 Windows 系统为操作平台。为便于查询和保存数据，还需要数据库的支持，这种应用的核心是数据通信，它包括单片机和上位机之间、客户端和服务端之间以及客户端和客户端之间的通信，而单片机和上位机之间数据通信则是整个系统的基础。

单片机和 PC 的通信是通过单片机的串口和 PC 串口之间的硬件连接实现的。

图 13-11 所示是本设计使用的单片机实验板。有关单片机实验板的详细信息请查询电子开发网 <http://www.dzkw.com/>。

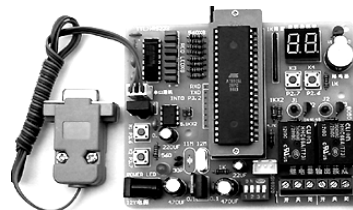


图 13-11 单片机实验板

13.2.1 PC 与单片机串口通信程序设计硬件线路

如图 13-12 所示，数据通信的硬件上采用 3 线制，将单片机和 PC 串口的 3 个引脚（RXD、TXD、GND）分别连在一起，即将 PC 和单片机的发送数据线 TXD 与接收数据 RXD 交叉连接，两者的地线 GND 直接相连，而其他信号线，如握手信号线均不用，采用软件握手的方式，这样既可以实现预定的任务又可以简化电路设计。

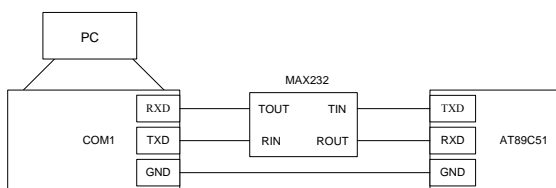


图 13-12 PC 与单片机串口通信线路

但由于单片机的 TTL 逻辑电平和 RS-232C 的电气特性完全不同，RS-232C 的逻辑 0 电平规定为 +3V~+15V 之间，逻辑 1 电平为 -3V~-15V 之间，因此在将 PC 和单片机的 RXD 和 TXD 交叉连接时必须进行电平转换，这里使用的是 MAX232 电平转换芯片。

单片机系统有 LED 显示器模块、继电器输出模块、蜂鸣器模块等。

13.2.2 PC 与单片机串口通信程序设计任务

利用 Keil C51 和 LabVIEW 编写程序实现 PC 与单片机串口通信。

任务要求有以下几方面。

1. 设计任务一

PC 通过串行口将数字（00, 01, 02, 03..., FF，十六进制）发送给单片机，单片机收到后回传这个数字，PC 接收到回传数据后显示出来，若发送的数据和接收到的数据相等，则串行通信正确，否则有错误。起始符是数字 00，结束符是数字 FF。

2. 设计任务二

(1) 测试通信状态。

先在文本框中输入字符串“Hello”，单击“测试”按钮，将字符串“Hello”发送到单片机，若 PC 与单片机通信正常，在 PC 程序的文本框中显示字符串“OK!”，否则显示字符串“ERROR!”。

(2) 循环计数。

单击“开始”按钮，文本框中数字从 0 开始累加，0、1、2、3...，并将此数发送到单片机

的显示器上显示。当累加到 10 时，回到 0 重新开始累加，依次循环。任何时候，单击“停止”按钮，PC 程序中和单片机显示器都停止累加；再单击“开始”按钮，接着停下的数继续累加。

(3) 控制指示灯。

在单片机继电器接线端子的两个通道上分别接上两个指示灯，在 PC 程序画面上选择指示灯号，如 1 号灯，单击画面“打开”按钮，单片机上 1 号灯亮，同时蜂鸣器响；单击画面“关闭”按钮，1 号灯灭，蜂鸣器停止响，同样控制 2 号灯的亮灭（蜂鸣器同时动作）。

单片机和 PC 通信，在程序设计上涉及两个部分的内容。

一是单片机的 C51 程序，二是 PC 的串口通信程序和界面的编制。

13.2.3 任务实现

13.2.3.1 利用 Keil C51 实现单片机与 PC 串口通信任务一

Keil C51 软件是众多单片机应用开发的优秀软件之一，它集编辑、编译、仿真于一体，支持汇编、PLM 语言和 C 语言的程序设计，界面友好，易学易用。

启动 Keil C51，出现编辑界面。

1. 建立一个新工程

单击 Project 菜单，在弹出的下拉菜单中选中 New Project 选项，出现 Create New Project 对话框，然后选择要保存的路径、文件夹，输入工程文件的名字，如 pc_com（后缀名默认），单击“保存”按钮。

这时会弹出一个“Select Device for Target ‘Target 1’”对话框，要求用户选择单片机的型号，可以根据使用的单片机来选择，Keil C51 几乎支持所有的 51 核的单片机。这里选择 Atmel 的 89C51。选择 89C51 之后，右边一栏是对这个单片机的基本的说明，然后单击“确定”按钮。

2. 编写程序

单击“File”菜单，再在下拉菜单中单击“New”选项。此时光标在编辑窗口里闪烁，这时可以键入用户的应用程序了，但建议首先保存该空白的文件。

单击菜单上的“File”项，在下拉菜单中选中“Save As”选项，在“文件名”栏右侧的编辑框中键入欲使用的文件名，同时，必须键入正确的扩展名，如 pc_com.c，然后单击“保存”按钮。



如果用 C 语言编写程序，则扩展名为 (.c)；如果用汇编语言编写程序，则扩展名必须为 (.asm)。

回到编辑界面后，单击“Target 1”前面的“+”号，再在“Source Group 1”上单击鼠标右键，弹出快捷菜单，然后单击“Add File to Group ‘Source Group 1’”。

选中 pc_com.c，然后单击“Add”按钮，再单击“Close”按钮。此时注意到“Source Group 1”文件夹中多了一个子项“pc_com.c”。子项的多少与所增加的源程序的多少相同。

现在，请输入 C 语言源程序。

在输入程序时，读者可以发现事先保存待编辑的文件的好处，即 Keil C51 会自动识别关键字，并以不同的颜色提示用户加以注意，这样会使用户少犯错误，有利于提高编程效率。

3. 编译程序

单击“Project”菜单，在下拉菜单中选择“Options for Target ‘Target 1’”选项，出现对话框；选择 Output 选项卡，选中“Create HEX Files”项，单击“确定”按钮。

再单击“Project”菜单，在下拉菜单中选择“Built Target”选项（或者使用快捷键 F7），进行编译。若有错误会在 output 窗口提示，可根据此提示，找出错误并修改，直至编译通过，如图 13-13 所示。

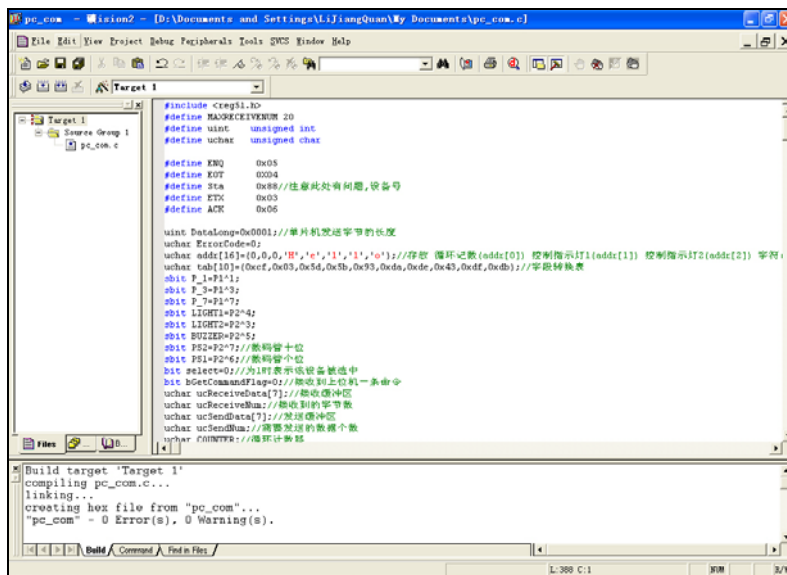


图 13-13 Keil C51 编译界面

至此，用 Keil C51 上做了一个完整工程，其中，生成一个编程器烧写文件 pc_com.hex。

4. 烧录程序

将 AT89C51 芯片安装在编程器插座上，运行编程器程序。

选择单片机芯片类型 AT89C51，读入 Intel Hex 文件 pc_com.hex，执行自动编程指令，将 pc_com.hex 文件烧录入 AT89C51 芯片中。

将烧录好的 AT89C51 芯片安装到单片机实验板上，就可以用串口调试助手程序对它进行测试了。

以下是完成单片机与 PC 串口通信任务 1 的 C51 参考程序：

```
# pragma db code
# include<reg51.h>
# define uchar unsigned char

void rece(void);
void init(void);
uchar re[17];

/*主程序*/
void main(void)
```

```

{
uchar temp;
init();
do{
    while(RI==0);
    temp=SBUF;
    if(temp==0x00)
        {rece();}
    else break;
}while(1);
}
/*串口初始化*/
void init(void)
{
    TMOD=0x20;           //定时器 1--方式 2
    PCON=0x80;           //电源控制
    SCON=0x50;           //方式 1
    TL1=0xF3;
    TH1=0xF3;           //22.1184MHz 晶振, 波特率为 4800 0xf3   9600   0xfa   19200 0xfd
    TR1=1;
}
/*接收返回数据*/
void rece(void)
{
    char i;
    i=0;
    do{while(RI==0);
        re[i]=SBUF;
        RI=0;
        SBUF=re[i];
        while(TI==0);
        TI=0;
        i++;
    }while(re[i-1]!=255);
}

```

13.2.3.2 利用 LabVIEW 实现 PC 与单片机串口通信任务一

1. 建立新 VI 程序

启动 NI LabVIEW 程序, 选择新建 (New) 选项中的 VI 项, 建立一个新 VI 程序。

2. 程序前面板设计

☞ 在前面板设计区空白处单击鼠标右键，显示控件选板（Controls）。

(1) 添加一个字符串输入控件：控件（Controls）→新式（Modern）→字符串与路径（String & Path）→字符串输入控件（String Control），将标签改为“发送数据（十六进制）”，在该控件上单击鼠标右键，在弹出的快捷菜单中选择“十六进制显示（Hex Display）”。

(2) 添加一个字符串显示控件：控件（Controls）→新式（Modern）→字符串与路径（String & Path）→字符串显示控件（String Indicator），将标签改为“返回数据（十六进制）”，在该控件上单击鼠标右键，在弹出的快捷菜单中选择“十六进制显示（Hex Display）”。

(3) 添加一个字符串显示控件：控件（Controls）→新式（Modern）→字符串与路径（String & Path）→字符串显示控件（String Indicator），将标签改为“通信状态”。

(4) 添加一个串口资源检测控件：控件（Controls）→新式（Modern）→I/O → VISA 资源名称（VISA resource name）；单击控件箭头，选择串口号，如 ASRL1:或 COM1。

(5) 添加一个确定按钮控件：控件（Controls）→新式（Modern）→布尔（Boolean）→确定按钮（OK Butoon），将标题改为“发送”。

(6) 添加一个停止按钮控件：控件（Controls）→新式（Modern）→布尔（Boolean）→停止按钮（Stop Butoon），将标题改为“关闭”。



图 13-14 程序前面板

3. 框图程序设计——添加函数与连线

☞ 进入框图程序设计界面，在设计区的空白处单击鼠标右键，显示函数选板（Functions）。

(1) 添加一个配置串口函数：编程（Programming）→仪器 I/O（Instrument I/O）→串口（Serial）→VISA 配置串口（VISA Configure Serial Port）。

(2) 添加 4 个数值常量：编程（Programming）→数值（Numeric）→数值常量（Numeric Constant），值分别为 4800（波特率）、8（数据位）、0（校验位，无）、1（停止位）。

(3) 添加一个 While 循环结构：编程（Programming）→结构（Structures）→While 循环（While Loop）。

(4) 添加一个关闭串口函数：编程（Programming）→仪器 I/O（Instrument I/O）→串口（Serial）→VISA 关闭（VISA Close）。

(5) 在 While 循环结构中添加一个条件结构：编程（Programming）→结构（Structures）→条件结构（Case Structure）。

(6) 在条件结构中添加一个顺序结构：编程（Programming）→结构（Structures）→层叠式顺序结构（Stacked Sequence Structure）。

将其帧（Frame）设置为 4 个（序号 0-3）。设置方法：选中 Stacked Sequence Structures 上边框，单击鼠标右键，执行在后面添加帧（Add Frame After）选项 3 次。




(7) 在顺序结构的 Frame 0 中添加一个串口写入函数：编程（Programming）→仪器 I/O（Instrument I/O）→串口（Serial）→VISA 写入（VISA Write）。

(8) 将控件“发送数据（十六进制）”的图标拖入顺序结构的 Frame 0 中，分别将确定按

钮 (OK Button)、停止按钮 (Stop Buffon) 的图标拖入循环结构中。

(9) 将 VISA 资源名称 (VISA resource name) 函数的输出端口分别与串口配置 (VISA Configure Serial Port) 函数、串口写入 (VISA Write) 函数 (在顺序结构 Frame 0 中)、串口关闭 (VISA Close) 函数的输入端口 VISA 资源名称 (VISA resource name) 相连。

(10) 将数值常量 4800、8、0、1 分别与 VISA 配置串口 (VISA Configure Serial Port) 函数的输入端口波特率 (baud rate)、数据位 (data bits)、奇偶 (parity)、停止位 (stop bits) 相连。

(11) 右键选择循环结构的条件端子 , 设置为“真时停止 (Stop if True)”, 图标变为 。将停止按钮 (Stop Buffon) 与循环结构的条件端子  相连。

(12) 将确定按钮 (OK Button) 与条件结构的选择端子? 相连。

(13) 将函数“发送数据 (十六进制)”与串口写入 (VISA Write) 函数的输入端口写入缓冲区 (write buffer) 相连。

连接好的框图程序如图 13-15 所示。

(14) 在顺序结构的 Frame 1 中添加一个时钟函数: 编程 (Programming) → 定时 (Timing) → 等待下一个整数倍毫秒 (Wait Until Next ms Multiple)。

(15) 在顺序结构的 Frame 1 中添加一个数值常量: 编程 (Programming) → 数值 (Numeric) → 数值常量 (Numeric Constant), 将值改为 200 (时钟频率值)。

(16) 在顺序结构的 Frame 1 中将数值常量 (值为 200) 与等待下一个整数倍毫秒 (Wait Until Next ms Multiple) 函数的输入端口毫秒倍数 (millisecond multiple) 相连。

连接好的框图程序如图 13-16 所示。

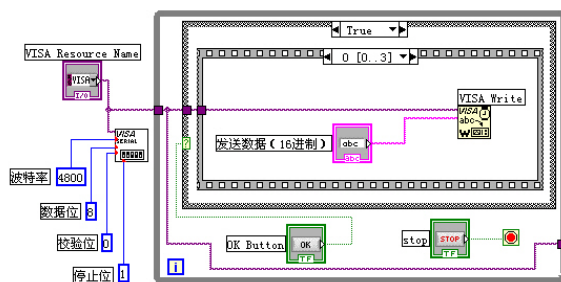


图 13-15 框图程序连线 1

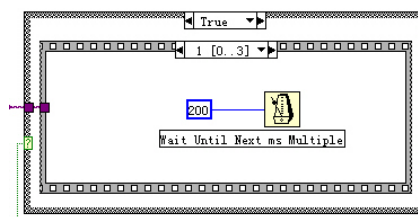


图 13-16 框图程序连线 2

(17) 在顺序结构的 Frame 2 中, 添加一个串口字节数函数: 编程 (Programming) → 仪器 I/O (Instrument I/O) → 串口 (Serial) → VISA 串口字节数 (VISA Bytes at Serial Port), 标签为“Property Node”。

(18) 在顺序结构的 Frame 2 中, 添加一个串口读取函数: 编程 (Programming) → 仪器 I/O (Instrument I/O) → 串口 (Serial) → VISA 读取 (VISA Read)。

(19) 将控件“返回数据 (十六进制)”的图标拖入顺序结构的 Frame 2 中。

(20) 将 VISA 串口字节数 (VISA Bytes at Serial Port) 函数的输出端口 VISA 资源名称 (VISA resource name) 与 VISA 读取 (VISA Read) 函数的输入端口 VISA 资源名称 (VISA resource name) 相连。

(21) 将 VISA 串口字节数 (VISA Bytes at Serial Port) 函数的输出端口 Number of bytes at Serial port 与串口读取 (VISA Read) 函数的输入端口字节总数 (byte count) 相连。

(22) 将 VISA 读取 (VISA Read) 函数的输出端口读取缓冲区 (read buffer) 与控件“返

回数据（十六进制）”的输入端口相连。

连接好的框图程序如图 13-17 所示。

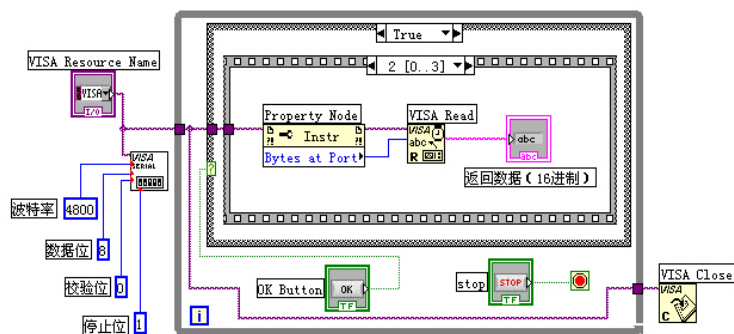


图 13-17 框图程序连线 3

(23) 在顺序结构的 Frame 3 中，添加两个局部变量：编程 (Programming) → 结构 (Structures) → 局部变量 (Local Variable)。

选择局部变量，单击鼠标右键，在弹出的快捷菜单的 (Select Item) 选项下，为局部变量分别选择对象：“返回数据（十六进制）”和“发送数据（十六进制）”，将其读写属性设置为“转换为读取 (Change To Read)”。

(24) 在顺序结构的 Frame 3 中，添加一个比较函数：编程 (Programming) → 比较 (Comparison) → 等于? (Equal?)。

(25) 在顺序结构的 Frame 3 中，添加一个条件结构：编程 (Programming) → 结构 (Structures) → 条件结构 (Case Structure)。

(26) 将局部变量“返回数据（十六进制）”和“发送数据（十六进制）”分别与比较函数等于? (Equal?) 的输入端口 x 和 y 相连。

(27) 将比较函数等于? (Equal?) 的输出端口 $x=y?$ 与条件结构的选择端子? 相连。

(28) 在条件结构的真 (True) 选项卡中，添加一个字符串常量：编程 (Programming) → 字符串 (String) → 字符串常量 (String Constant)，将其值改为“通信正常!”。

(29) 将控件“通信状态”拖入条件结构中。

(30) 将字符串常量“通信正常!”与控件“通信状态”的输入端口相连。

(31) 在条件结构的假 (False) 选项卡中，添加一个字符串常量，将其值改为“通信异常!”。

(32) 在条件结构的假 (False) 选项卡中，添加一个局部变量，为局部变量选择对象“通信状态”，属性默认为：“写”。

(33) 将字符串常量“通信异常!”与局部变量“通信状态”相连。

连接好的框图程序如图 13-18 所示。

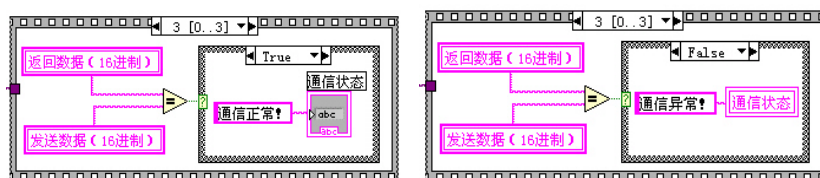


图 13-18 框图程序连线 4

4. 运行程序

进入程序前面板，保存设计好的 VI 程序。单击快捷工具栏中的“运行 (Run)”按钮，运行程序。程序运行界面如图 13-19 所示。

在“发送数据”框中输入两位的十六进制数字 (00, 01, 02, 03..., FF)，单击“发送”按钮，将数据发送给单片机；单片机收到后回传这个数字，PC 接收到回传数据后在“返回数据”框中显示出来 (十六进制)，若发送的数据和接收到的数据相等，则在“通信状态”框中显示“通信正常！”，否则显示“通信异常！”。

当发送“FF”后，要想继续发送数据，必须先发送“00”。

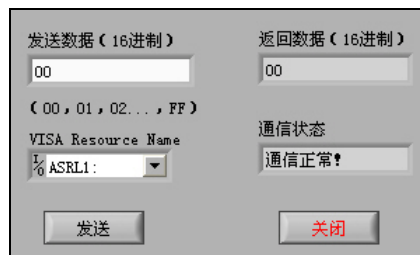


图 13-19 程序运行界面

13.2.3.3 利用 Keil C51 实现单片机与 PC 串口通信任务二

```
.....
}
```

13.2.3.4 利用 LabVIEW 实现 PC 与单片机串口通信任务二

```
.....
```

13.3 PC 与智能仪器串口通信案例

目前仪器仪表的智能化程度越来越高，大量的智能仪器都配备了 RS-232 通信接口，并提供了相应的通信协议，能够将测试、采集的数据传输给计算机等设备，以便进行大量数据的储存、处理、查询和分析。通常计算机 (PC) 或工控机 (IPC) 是智能仪器上位机的最佳选择，因为 PC 或 IPC 不仅能解决智能仪器 (作为下位机) 所不能解决的问题，如数值运算、曲线显示、数据查询、报表打印等；而且具有丰富和强大的软件开发工具环境。

图 13-23 是 XMT-3000A 型智能仪器示意图 (详细信息请查询网站 <http://www.njcy.com/>)。



图 13-23 智能仪器示意图

13.3.1 PC 与智能仪器串口通信硬件线路

1. 线路说明

XMT-3000A 智能仪器采用先进的微电脑芯片、专家 PID 控制算法，具备高准确度的自整定功能，并可以设置出多种报警方式。

XMT-3000A 智能仪表有多种输入功能，一台仪表可以接热电偶（K、S、Wr、E、J、T、B、N）、热电阻（Pt100、Cu50）、电压（0~5V、1V~5V）、电流（0~10mA、4mA~20mA）等不同的输入信号。

XMT-3000A 智能仪表接热电阻输入时，采用三线制接线，消除了引线带来的误差；接热电偶输入时，仪表内部带有冷端补偿部件；接电压/电流输入时，对应显示的物理量程可任意设定。

在计算机与智能仪器通电前，按图 13-24 所示将热电阻传感器 Cu50、上、下限报警指示灯与 XMT-3000A 智能仪器连接。

通过串口线将计算机与智能仪器连接起来：智能仪器的 14 端子（RXD）与计算机串口 COM1 的 3 脚（TXD）相连；智能仪器的 15 端子（TXD）与计算机串口 COM1 的 2 脚（RXD）相连；智能仪器的 16 端子（GND）与计算机串口 COM1 的 5 脚（GND）相连。

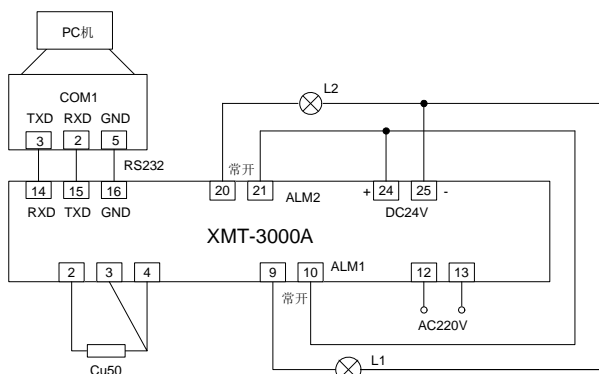


图 13-24 PC 与智能仪表串口通信线路



连接仪器与计算机串口线时，仪器与计算机严禁通电，否则极易烧毁串口。

本设计用到的硬件为：智能仪器（XMT-3000A 型，需配置 RS-232 通信、上下限控制继电器、DC24V 电源等模块），串口通信线（三线制），热电阻传感器（Cu50），指示灯（DC24V）等。

2. XMT-3000A 智能仪器的参数设置

XMT-3000A 智能仪器在使用前应对其输入/输出参数进行正确设置，设置好的仪器才能投入正常使用。

请按表 13-1 设置仪器的主要参数。

表 13-1

仪表的主要参数设置

参 数	参 数 含 义	设 置 值
HiAL	上限绝对值报警值	30
LoAL	下限绝对值报警值	20
Sn	输入规格	传感器为: Cu50, 则 Sn=20
diP	小数点位置	要求显示一位小数, 则 diP=1
ALP	仪表功能定义	要求上限报警由报警 1 (ALM1) 输出, 下限报警由报警 2 (ALM2) 输出, 报警时在下显示器显示报警符号, 则 ALP=10
Addr	通信地址	0
baud	通信波特率	4800

3. XMT-3000A 智能仪表的通信协议

XMT-3000A 智能仪器使用异步串行通信接口, 共有两种通信方式: RS232C 和 RS485。接口电平符合 RS232C 或 RS485 标准中的规定。数据格式为一个起始位, 8 位数据, 无校验位, 2 个停止位。通信传输数据的波特率可调为 300~4800 bit/s。

XMT 仪表采用多机通信协议, 如果采用 RS485 通信接口, 则可将 1~64 台的仪表同时连接在一个通信接口上; 采用 RS232C 通信接口时, 一个通信接口只能连接一台仪表。

RS485 通信接口与 RS422 接口的信号电平相同, 通信距离长达 1km 以上, 优于 RS232C 通信接口。RS422 为全双工工作方式, RS485 为半双工工作方式, RS485 只需两根线就能使多台 XMT 仪表与计算机进行通信, 而 RS422 需要 4 根通信线。由于通信协议的限制, XMT 只能工作在半双工模式, 所以 XMT 仪表推荐使用 RS485 接口, 以简化通信线路接线。为使普通计算机作上位机, 可使用 RS232C/RS485 型通信接口转换器, 将计算机上的 RS232C 通信口转为 RS485 通信口。

XMT 仪表采用十六进制数据格式来表示各种指令代码及数据。

通信指令只有两条, 一条为读指令, 一条为写指令。

读指令格式: 地址代号+52H+参数代号。

返回: 依次返回为测量值 PV、给定值 SV、输出值 MV+报警状态、所读参数值。

写指令格式: 地址指令+43H+参数代号+写入值的低位字节+写入值的高位字节。

返回: 测量值 PV、给定值 SV、输出值 MV+报警状态、被写入的参数值。

地址代号: 为了在一个通信接口上连接多台 XMT 仪表, 需要给每台 XMT 仪表编一个互不相同的代号, 这一代号在本文约定称为通信地址代号 (简称地址代号)。XMT 有效的地址为 0~63。所以一条通信线路上最多可连接 64 台 XMT 仪表。仪表的地址代号由参数 Addr 决定。

XMT 仪表通信协议规定, 地址代号为两个字节, 其数值范围 (十六进制) 是 80H~BFH, 两个字节必须相同, 数值为: 仪表地址+80H。例如, 仪表参数 Addr=5 (十六进制数为 05H), 05+80H=85H, 则该仪表的地址表示为: 85H 85H。

参数代号: 仪表的参数用一个十六进制数的参数代号来表示。它在指令中表示要读/写的参数名。表 13-2 列出了 XMT 仪表可读/写的参数代号 (部分)。

表 13-2 XMT 仪表可读/写的参数代号表

参 数 代 号	参 数 名	含 义	参 数 代 号	参 数 名	含 义
00H	SV	给定值	0BH	Sn	输入规格
01H	HiAL	上限报警值	0CH	dIP	小数点位置
02H	LoAL	下限报警值	0DH	dIL	下限显示值
03H	dHAL	正偏差报警	0EH	dIH	上限显示值
04H	dLAL	负偏差报警	15H	baud	通信波特率
05H	dF	回差	16H	Addr	通信地址
06H	Ctrl	控制方式	17H	dL	数字滤波



如果向仪表读取参数代号在表格中参数以外, 则返回参数值为错误信号 (两个 7F 值)。

返回的测量值数据每两个 8 位数据代表一个 16 位整形数, 低位字节在前, 高位字节在后, 负温度值采用补码表示, 热电偶或热电阻输入时其单位都是 0.1°C , 回送的十六进制数据 (两个字节) 先转换为十进制数据, 然后将十进制数据除以 10 再显示出来。

上位机每次向仪表发一个指令, 仪表返回一个数据。编写上位机软件时, 注意每条有效指令, 仪表在 $0\sim 0.36\text{s}$ 内作出应答, 而上位机也必须等仪表返回指令后, 才能发新的指令, 否则将引起错误。

4. 温度测量与控制

(1) 正确设置仪器参数后, 仪器 PV 窗显示当前温度测量值。

(2) 给传感器升温, 当温度测量值大于上限报警值 30°C 时, 上限指示灯 L2 亮, 仪器 SV 窗显示上限报警信息。

(3) 给传感器降温, 当温度测量值小于上限报警值 30°C , 大于下限报警值 20°C 时, 上限指示灯 L2 和下限指示灯 L1 均灭。

(4) 给传感器继续降温, 当温度测量值小于下限报警值 20°C 时, 下限指示灯 L1 亮, 仪器 SV 窗下限报警信息。

5. 串口调试

XMT-3000A 智能仪器使用异步串行通信接口, 采用 RS-232 通信方式, 其数据格式为: 一个起始位, 8 个数据位, 无校验位, 2 个停止位。

打开“串口调试助手”程序, 首先设置串口号 COM1、波特率 4800、校验位 NONE、数据位 8、停止位 2 等参数 (注意: 设置的参数必须与仪器设置的一致), 选择十六进制显示和十六进制发送方式, 打开串口, 如图 13-25 所示。

在“发送的字符/数据”文本框中输入读指令: 80 80 52 0C, 单击“手动发送”按钮, 则 PC 向仪器发送一条指令, 仪器返回一串数据, 如: 3D 01 E7 03 64 00 01 00, 该串数据在返回信息框内显示。

根据仪器返回数据, 可知仪器的当前温度测量值为: 01 3D (十六进制, 低位字节在前, 高位字节在后), 十进制为 31.7°C 。

使用说明。

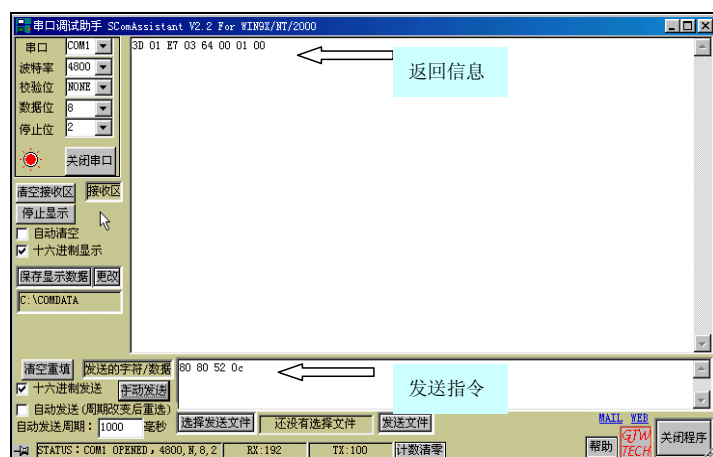


图 13-25 串口调试助手

若选择了“手动发送”，每单击一次可以发送一次；若选中了“自动发送”，则每隔设定的发送周期内发送一次，直到去掉“自动发送”为止。值得注意的一点是：选中“十六进制发送”后，发送框中所填字符每两个字符之间应有一个空格，如：01 23 00 34 45。还有一些特殊的字符，如回车换行，则直接敲入回车即可。

6. 使用“计算器”实现数制转换

打开 Windows 附件中“计算器”程序，在“查看”菜单下选择“科学型”。

选择“十六进制”，输入仪器当前温度测量值：01 3D（十六进制，0 在最前面不显示），如图 13-26 所示。

单击“十进制”选项，则十六进制数“013D”转换为十进制数“317”，如图 13-27 所示。仪器的当前温度测量值为：31.7°C（十进制）。为什么？



图 13-26 在“计算器”中输入十六进制数

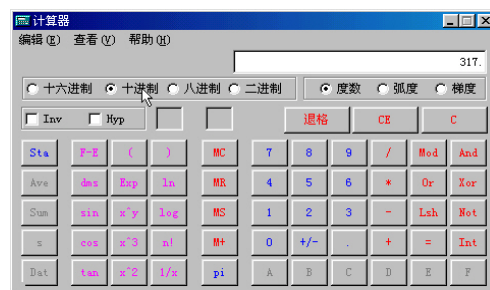


图 13-27 十六进制数转十进制数

13.3.2 设计任务

.....

13.4 PC 与 PLC 串口通信案例

可编程序逻辑控制器（简称 PLC）主要是为现场控制而设计的，其人机界面主要是开关、按钮、指示灯等。其具有良好的适应性和可扩展能力得到越来越广泛的应用。采用 PLC 的控制系统或装置具有可靠性高、易于控制、系统设计灵活、能模拟现场调试、编程使用简单、性价比高、有良好的抗干扰能力等特点。但是，PLC 也有不易显示各种实时图表/曲线（趋势线）和汉字、无良好的用户界面、不便于监控等缺陷。

20 世纪 90 年代后，许多的 PLC 都配备有计算机通信接口，通过总线将一台或多台 PLC 相连接。计算机作为上位机可以提供良好的人机界面，进行系统的监控和管理，进行程序编制、参数设定和修改、数据采集等，既能保证系统性能，又能使系统操作简便，便于生产过程的有效监督。而 PLC 作为下位机，执行可靠有效的分散控制。用一台计算机（上位机）去监控下位机（PLC），这就要求 PC 与 PLC 之间稳定、可靠的数据通信。



图 13-34 是某型号 PLC 示意图。

图 13-34 PLC 产品示意图

13.4.1 PC 与 PLC 串口通信硬件线路

西门子 S13-200PLC 系统为用户提供了灵活的通信功能。集成在 S13-200 中的点对点接口（PPI）可用普通的双绞线作波特率高达 9600bit/s 的数据通信，用 RS-485 接口实现的高速用户可编程接口，可使用专用位通信协议（如 ASCII）做波特率高达 38.4 kbit/s 的高速通信并可按步调整。而 PC 的接口为 RS-232，两者之间需要进行电平转换。利用西门子公司的 PC/PPI 电缆，可将 S13-200CPU 与计算机连接起来组成 PC/PPI 网络，实现点对点通信，如图 13-35 所示。

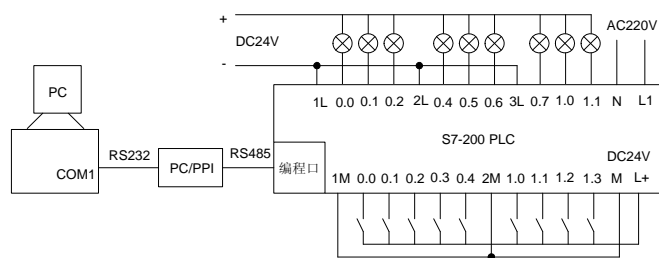


图 13-35 PC 与 S13-200PLC 串口通信线路

本设计用到的硬件为：西门子 S13-200PLC（CPU224），PC/PPI 电缆，开关，指示灯（DC24V），直流电源（OUT：DC24V）等。

13.4.2 设计任务

利用 LabVIEW 编写程序实现 PC 与 PLC 串口通信。

任务要求。

（1）开关量输入：利用继电器开关改变某个输入端口的状态，程序读取该端口的输入状态（打开/关闭），并在程序中显示。

(2) 开关量输出：程序画面中指定元件地址，单击置位/复位命令按钮，置指定地址的元件端口（继电器）状态为 ON 或 OFF，使线路中指示灯亮/灭。

。 。 。 。 。

13.5 PC 与 GSM 短信模块串口通信案例

在很多监控领域，各种监控设备大多还是有线方式传输，当距离遥远时，监控设备的安装、维护非常不便，因为监控端远离采集端，铺设电缆的投入有时可以说是巨大的。通过无线方式来交换数据，则可以有效地避免这些问题。

GSM 网络是目前国内覆盖范围最广，应用最普遍的无线网络，利用 GSM 网络构建远程监测系统时，完全可利用现成的 GSM 无线网络而无需再新建基站。

利用 GSM 网络短消息业务（SMS）实现监测领域的应用具有以下优点：通信网络覆盖面广、网络设施完备，不需投资建设基础设施；实施与运行费用低；可以实现在无人职守、环境恶劣、超远距离的情况下控制信息的收集和传送。

GSM 短信模块是专门用于短信接收发送的模块，具有 RS-232 通信口，可与单片机、计算机相连。

图 13-38 所示为某型号 GSM 模块示意图。



图 13-38 GSM 模块示意图

13.5.1 PC 与 GSM 短信模块串口通信硬件线路

在数据采集站，传感器检测的数据送入单片机模拟量输入口，单片机通过串口与 GSM 模块相连；在监控中心，GSM 模块通过串口线与 PC 直接相连，如图 13-39 所示。

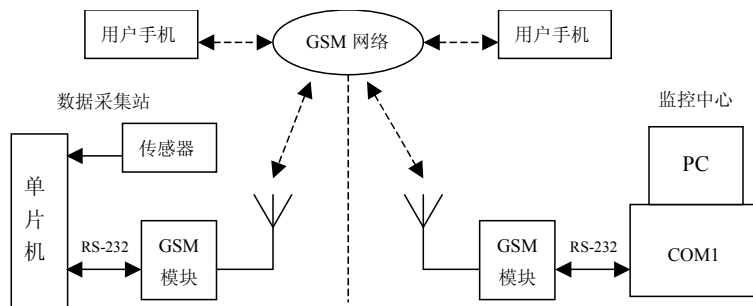


图 13-39 利用 GSM 模块组成的远程监控系统

传感器检测的数据经单片机 MCU 单元的处理，编辑成短信息，通过串行口传送给 GSM 模块后以短消息的方式将数据发送到监控中心的计算机或用户的 GSM 手机。

同样，监控中心 PC 通过串口向 GSM 模块 TC35 发送命令，通过 TC35 以短消息形式把设置命令发送到下位机系统的 GSM 模块，对单片机进行控制。

本设计用到的硬件为：GSM 短信模块（TC35i），串口通信线（三线制），SIM 卡，手机等。用户手机通过 GSM 模块与 PC 和单片机可以实现双向通信。

13.5.2 PC 与 GSM 短信模块串口通信设计任务

利用 LabVIEW 编写程序实现 PC 与 GSM 短信模块串口通信。

任务要求。

- (1) 在程序画面输入短信内容，指定接收方手机号码，将编辑的短信息发送到用户手机。
- (2) 用户手机向监控中心的 GSM 模块发送短信，程序界面显示短信内容及来电号码。

13.5.3 任务实现

.....

13.6 PC 与智能仪器构成 DCS 案例

智能仪器在我国的工业控制领域得到了广泛的应用。实际上，只要具有 RS-485（或 RS-232）通信接口、支持站号设置和通信协议访问的智能仪器都可以和 PC 构成一个主从式网络系统，这也是中小型 DCS 的一般结构。智能仪器具有较强的过程控制功能和较高的可靠性，因此这类中小型 DCS 在目前仍然占有较大的应用市场。

13.6.1 PC 与智能仪器构成 DCS 硬件线路

1. 线路说明

由于一个 RS-232 通信接口只能联接一台 RS-232 仪表，当 PC 与多台具有 RS-232 接口的仪表通信时，可使用 RS-232/RS-485 型通信接口转换器，将计算机上的 RS-232 通信口转为 RS-485 通信口。在信号进入仪表前再使用 RS-485/RS-232 转换器将 RS-485 通信口转为 RS-232 通信口，再与仪表相连，如图 13-43 所示。

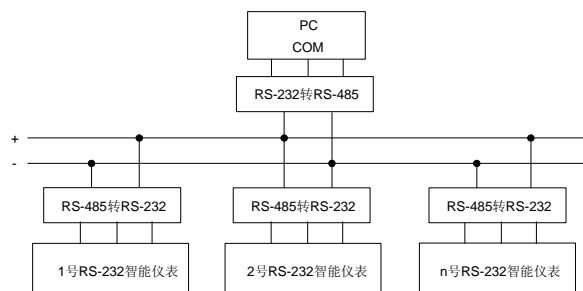


图 13-43 PC 与多个 RS-232 仪表连接示意图

当 PC 与多台具有 RS-485 接口的仪表通信时, 由于两端设备接口电气特性不一, 不能直接相连, 因此, 也采用 RS-232 接口到 RS-485 接口转换器将 RS-232 接口转换为 RS-485 信号电平, 再与仪表相连, 如图 13-44 所示。

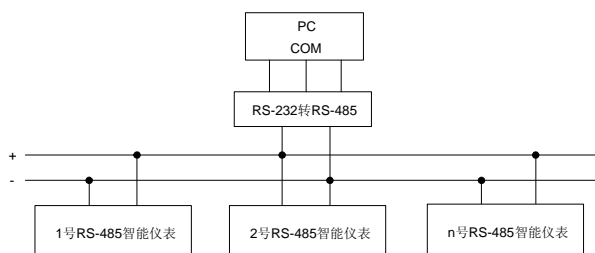


图 13-44 PC 与多个 RS-485 仪表连接示意图

如果 IPC 直接提供 RS-485 接口, 与多台具有 RS-485 接口的仪表通信时不用转换器可直接相连。

RS-485 接口只有两根线要连接, 有+、-端 (或称 A、B 端) 区分, 用双绞线将所有仪表的接口并联在一起即可。

本设计用到的硬件为: 3 台智能仪表 (XMT-3000A 型, 需配置 RS-232 通信、上下限控制继电器、DC24V 电源等模块), 3 个 RS-232/RS-485 转换器, 3 个热电阻传感器 (Cu50) 等。

2. XMT-3000A 智能仪表的参数设置

XMT-3000A 智能仪表在使用前应对其输入/输出参数进行正确设置, 设置好的仪表才能投入正常使用。请按表 13-3 设置仪表的主要参数。

表 13-3 XMT-3000A 智能仪表的参数设置

参 数	参 数 含 义	1 号仪表设置值	2 号仪表设置值	3 号仪表设置值
Sn	输入规格	20	20	20
diP	小数点位置	1	1	1
ALP	仪表功能定义	10	10	10
Addr	通信地址	1	2	3
bAud	通信波特率	4800	4800	4800

需要特别注意: DCS 系统中每台仪表有一个仪表号, PC 通过仪表号来识别网上的多台仪表, 要求网上的任意两台仪表的编号 (即地址代号 Addr 参数) 不能相同。所有仪表的波特率参数必须一样, 否则该地址的所有仪表通信都会失败。

3. 串口调试

运行“串口调试助手”程序, 首先设置串口号、波特率、校验位、数据位、停止位等参数 (与仪表参数设置一致), 选择十六进制显示和十六进制发送方式, 打开串口。

在发送指令文本框先输入读指令: 81 81 52 0C, 单击“手动发送”按钮, 1 号表返回数据串; 再输入读指令: 82 82 52 0C, 单击“手动发送”按钮, 2 号表返回数据串; 再输入读指令: 83 83 52 0C, 单击“手动发送”按钮, 3 号表返回数据串。

可用“计算器”程序分别计算各个表的测量温度值。

13.6.2 设计任务

利用 LabVIEW 编写程序实现 PC 与多个智能仪表串口通信。

- (1) 以十进制方式显示多个智能仪表温度测量值。
- (2) 读取并显示各个表的上、下限报警值。
- (3) 当测量温度值大于或小于上、下限报警值时，画面中相应的信号指示灯变化颜色。

13.6.3 任务实现

.....

13.7 PC 与远程 I/O 模块构成 DCS

远程 I/O 模块又称为牛顿模块，是近年来比较流行的一种 I/O 方式。它安装在工业现场，就地完成 A/D、D/A 转换、I/O 操作及脉冲量的计数、累计等操作。

远程 I/O 以通信方式和计算机交换信息，通信接口一般采用 RS-485 总线，通信协议与模块的生产厂家有关，但都是采用面向字符的通信协议。

市场上使用比较广泛的远程 I/O 模块有研华公司的 ADAM-4000 系列，如图 13-55 所示，以及研祥公司推出的 Ark-14000 系列等。这些远程 I/O 模块是传感器到计算机的多功能远程 I/O 单元，专为恶劣环境下的可靠操作而设计，具有内置的微处理器，严格的工业级塑料外壳，使其可以独立提供智能信号调理，模拟量 I/O、数字量 I/O，数据显示和 RS-485 通信。



图 13-55 远程 I/O 模块

13.7.1 硬件线路

1. 线路说明

如图 13-56 所示，ADAM-4520 与 PC 的串口 COM1 连接，并转换为 RS-485 总线；ADAM-4012 的 DATA+和 DATA-分别与 ADAM-4520 的 DATA+和 DATA-连接；ADAM-4050 的 DATA+和 DATA-分别与 ADAM-4520 的 DATA+和 DATA-连接。

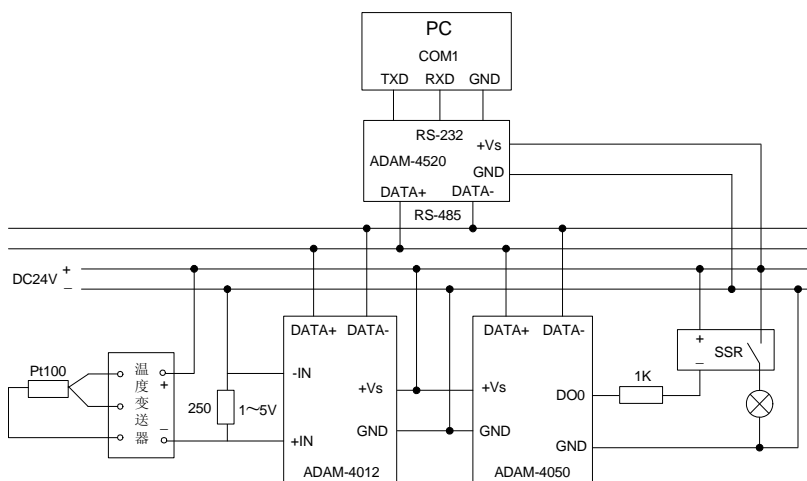


图 13-56 PC 与远程 I/O 模块串口通信线路

Pt100 热电阻检测温度变化，通过温度变送器转换为 4~20mA 电流信号，经过 250 Ω 电阻转换为 1~5V 电压信号送入 ADAM-4012 的模拟量输入通道。



变送器的“+”端接 24V 电源的高电压端（+），变送器的“-”端接模块的+IN，-IN 接 24V 电源低电压端（-）。

本设计用到的硬件为：研华公司的 ADAM-4520，ADAM-4012，ADAM-4050 模块，串口通信线（三线制），热电阻传感器（Pt100），温度变送器（输入：0~200℃，输出：4~20mA），直流电源（输出：DC24V）、固态继电器，电阻（250 Ω），电阻（1K），指示灯（DC24V）等。

2. 安装驱动程序

在使用研华 I/O 模块编程之前必须安装研华设备 DLL 驱动程序和设备管理程序 Device Manager。进入研华公司官方网站 www.advantech.com.cn，下载下列程序：ADAM_DLL.exe、DevMgr.exe、ADAM-4000-5000Utility.exe 等。

依次安装上述程序。

3. 模块配置

配置模块使用 Utility.exe 程序。运行 Utility.exe 程序，出现如图 13-57 所示的界面。

选中 COM1，单击工具栏中的快捷按钮 search，出现“Search Installed Modules”对话框，如图 13-58 所示。提示扫描模块的范围，允许输入 0~255，确定一个值后，单击“OK”按钮开始扫描。如果计算机 COM1 口安装有模块，将在程序右侧 COM1 下方出现已安装的模块名称，如图 13-59 所示。图 13-59 中显示 COM1 口安装了 4012 和 40502 个模块。

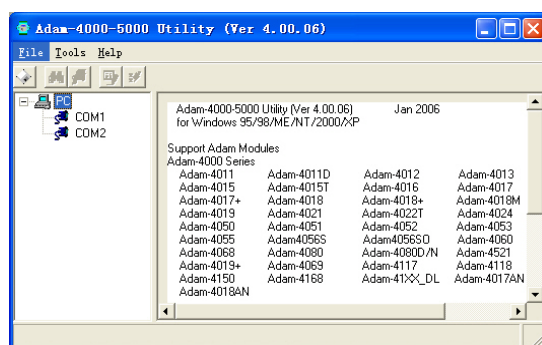


图 13-57 Utility 程序界面

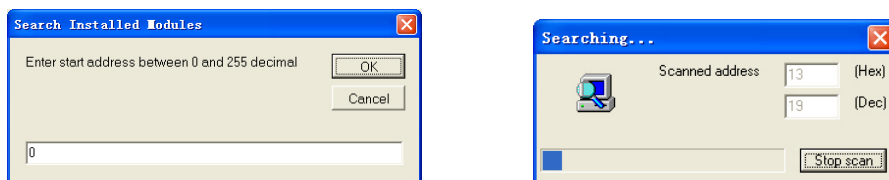


图 13-58 扫描安装的模块

单击模块名称“4012”，进入测试/配置界面，如图 13-60 所示。设置模块的地址值（1）、波特率（9600）、电压输入范围等，完成后，单击“Update”按钮。图 13-60 中模块名称 4012 前显示其地址值 01，AI 通道的输入电压是 1.4635V。

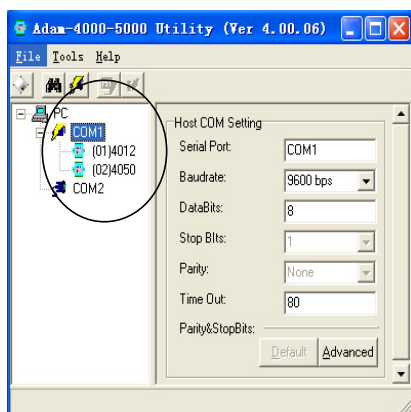


图 13-59 显示已安装的模块

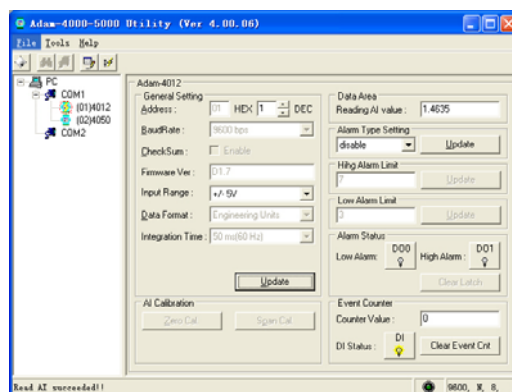


图 13-60 4012 模块配置与测试

单击模块名称“4050”，进入测试/配置界面，如图 13-61 所示。

设定波特率和校验和应注意：在同一条 485 总线上的所有模块和主计算机的波特率和校验和必须相同。连网前分别设置好两个模块的地址，不能重复。

4. 添加设备

运行设备管理程序 DevMgr.exe，在出现的对话框中从 Supported Devices 列表中选择“Advantech COM Devices”项，单击“Add”按钮，出现“Communication Port Configuration”对话框，设置串口通信参数，如图 13-62 所示。完成后，单击“OK”按钮。

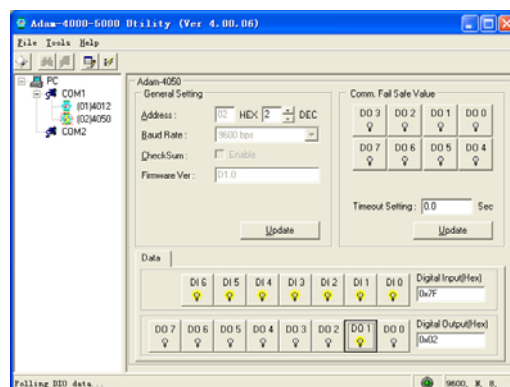


图 13-61 4050 模块配置与测试

展开“Advantech COM Devices”项，选择“Advantech ADAM-4000 Modules for RS-485”项，单击“Add”按钮，出现“Advantech ADAM-4000 Modules Parameters”对话框，如图 13-63 所示。在 Module Type 下拉列表中选择 ADAM 4012 在 Module Address 文本框中设置地址值，如 1（必须和模块的配置值一致）。

同样添加模块 ADAM 4050，地址值设为 2，完成后单击“OK”按钮。这时在 Installed Devices 列表中出现模块 ADAM 4012 与模块 ADAM 4050 的信息，如图 13-64 所示。



图 13-62 添加串口

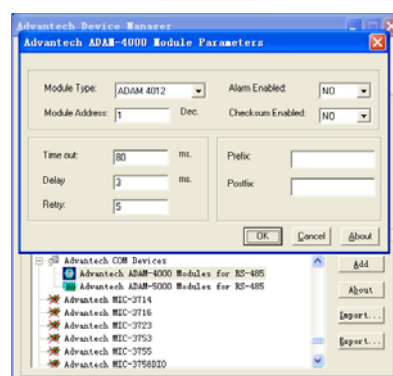


图 13-63 添加模块

在 Installed Devices 列表中选择模块“000 < ADAM 4012 Address=1 Dec.>”，单击右侧的“Test”按钮，出现“Advantech Devices Test”对话框，如图 13-65 所示。在 Analog Input 选项卡中，显示模拟输入电压值。图 13-65 中，ADAM-4012 模块的输入电压是 1.4235V。



图 13-64 模块添加完成

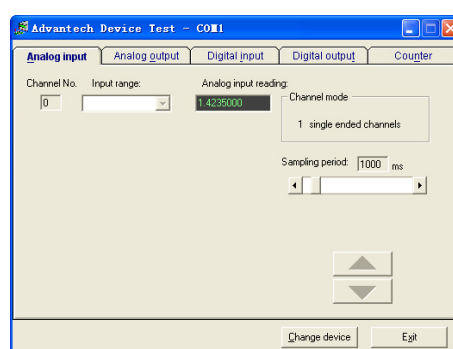


图 13-65 测试模块

至此，就可以用开发软件对 I/O 模块编程了。

13.7.2 设计任务

利用 LabVIEW 编写应用程序实现远程 I/O 模块温度测量与控制。

任务要求。

- (1) 自动连续读取并显示温度测量值；显示测量温度实时变化曲线。
- (2) 当测量温度大于设定值时，线路中指示灯亮。

13.7.3 任务实现

1. 建立新 VI 程序

启动 NI LabVIEW 程序，选择新建（New）选项中的 VI 项，建立一个新 VI 程序。

2. 程序前面板设计

在在前面板设计区的空白处单击鼠标右键，显示控件选板（Controls）。

(1) 添加一个仪表显示控件: 控件 (Controls) → 数值 (Numeric) → 仪表 (Meter), 将标签改为 “温度表”。

(2) 添加一个实时图形显示控件: 控件 (Controls) → 图形 (Graph) → 波形图形 (Waveform Chart), 将标签改为 “温度曲线”。

(3) 添加一个数值显示控件: 控件 (Controls) → 数值 (Numeric) → 数值显示控件 (Numeric Indicator), 将标签改为 “温度值:”。

(4) 添加一个指示灯控件: 控件 (Controls) → 布尔 (Boolean) → 圆形指示灯 (Round LED), 将标签分别改为 “指示灯”。

(5) 添加一个串口资源检测控件: 控件 (Controls) → 新式 (Modern) → I/O → VISA 资源名称 (VISA resource name); 单击控件箭头, 选择串口号, 如 ASRL1: 或 COM1。

(6) 添加一个停止按钮 (Stop Buffer) 控件: 控件 (Controls) → 布尔 (Boolean) → 停止按钮 (Stop Button), 标题为 “STOP”。

设计的程序前面板如图 13-66 所示。

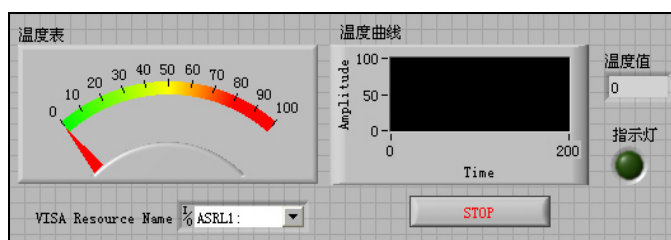


图 13-66 程序前面板

3. 框图程序设计——添加函数与连线

程序设计思路: 读温度值时, 向串口发送指令 “#01” + 回车, 模块向串口返回电压值 (字符串形式); 超温控制时, 向串口发送指令 “#021001” + 回车。

要解决两个问题: 如何发送读指令? 如何读取电压值并转换为数值形式?

① 进入框图程序设计界面, 在设计区空白处单击鼠标右键, 显示函数选板 (Functions)。

(1) 添加一个 While 循环结构: 编程 (Programming) → 结构 (Structures) → While 循环 (While Loop)。

(2) 在 While 循环结构中添加一个顺序结构: 编程 (Programming) → 结构 (Structure) → 层叠式顺序结构 (Stacked Sequence)。

将顺序结构的帧 (Frame) 设置为 5 个 (序号 0~4)。设置方法: 选中顺序结构 (Sequence), 单击鼠标右键, 执行在后面添加帧 (Add Frame After) 选项 4 次。


(3) 在顺序结构 Frame 0 中添加 5 个函数。

- 添加一个串口配置函数: 编程 (Programming) → 仪器 I/O (Instrument I/O) → 串口 (Serial) → VISA 配置串口 (VISA Configure Serial Port)。

- 添加 4 个数值常量: 编程 (Programming) → 数值 (Numeric) → 数值常量 (Numeric Constant), 值分别为 9600 (波特率)、8 (数据位)、0 (校验位, 无)、1 (停止位)。

- 将函数 VISA 资源名称 (VISA resource name) 的输出端口与串口配置 (VISA Configure Serial Port) 函数的输入端口 VISA 资源名称 (VISA resource name) 相连。

• 将数值常量 9600、8、0、1 分别与 VISA 配置串口 (VISA Configure Serial Port) 函数的输入端口波特率 (baud rate)、数据位 (data bits)、奇偶 (parity)、停止位 (stop bits) 相连。

(4) 在循环结构中的条件端口上单鼠标右键, 在弹出的快捷菜单上选择“真时停止 (Stop if True)”项; 将停止按钮图标与循环结构的条件端子  相连。

连接好的框图程序如图 13-67 所示。

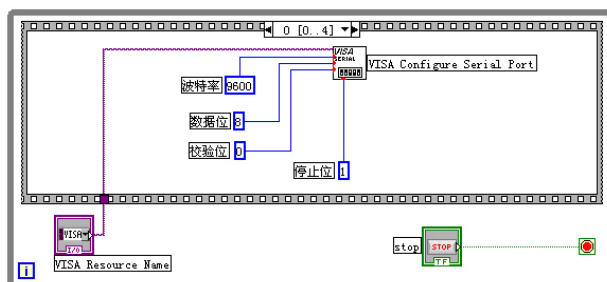


图 13-67 添加初始化串口函数

(5) 在顺序结构 Frame 1 中添加 4 个函数。

• 添加一个字符串常量: 编程 (Programming) → 字符串 (String) → 字符串常量 (String Constant), 值改为 “#01”, 标签为 “读 01 号模块 1 通道电压指令”。

• 添加一个回车键常量: 编程 (Programming) → 字符串 (String) → 回车键常量 (Carriage Return)。

• 添加一个字符串连接函数: 编程 (Programming) → 字符串 (String) → 连接字符串 (Concatenate Strings), 用于将读指令和回车符送给写串口函数。

• 添加一个串口写入函数: 编程 (Programming) → 仪器 I/O (Instrument I/O) → 串口 (Serial) → VISA 写入 (VISA Write)。

• 将字符串常量 (值为 “#01”, 读模拟量输入端口电压值的指令) 与连接字符串 (Concatenate Strings) 函数的输入端口 Substring 相连。

• 将回车键常量与连接字符串 (Concatenate Strings) 函数的另一个输入端口字符串 (Substring) 相连。

• 将连接字符串 (Concatenate Strings) 函数的输出端口连接的字符串 (Concatenated String) 与 VISA 写入 (VISA Write) 函数的输入端口写入缓冲区 (write buffer) 相连。

连接好的框图程序如图 13-68 所示。

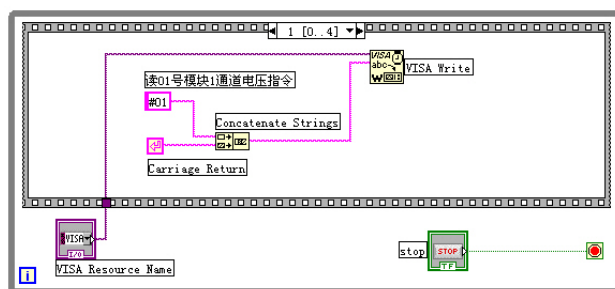


图 13-68 添加写指令函数

(6) 在顺序结构 Frame 2 中添加一个时间延迟函数: 编程 (Programming) → 定时 (Timing) → 时间延迟 (Time Delay), 时间采用默认值, 如图 13-69 所示。

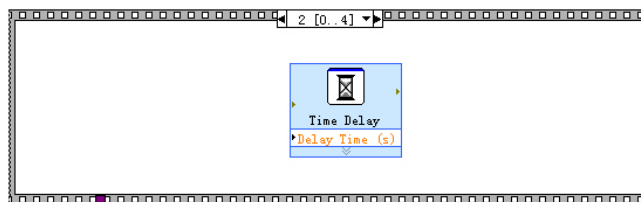


图 13-69 添加延时函数

(7) 在顺序结构 Frame 3 中添加函数。

- 添加一个串口字节数函数: 编程 (Programming) → 仪器 I/O (Instrument I/O) → 串口 (Serial) → VISA 串口字节数 (VISA Bytes at Serial Port), 标签为 “Property Node”。
- 添加一个串口读取函数: 编程 (Programming) → 仪器 I/O (Instrument I/O) → 串口 (Serial) → VISA 读取 (VISA Read)。
- 添加截取字符串函数: 编程 (Programming) → 字符串 (String) → 截取字符串 (String Subset)。
- 添加一个将字符串转换为数字函数: 编程 (Programming) → 字符串 (String) → 字符串/数值转换 (String /Number Conversion) → 分数/指数字符串至数值转换 (Fract/Exp String To Number)。
- 添加一个减号函数: 编程 (Programming) → 数值 (Numeric) → 减 (Subtract)。
- 添加一个乘号函数: 编程 (Programming) → 数值 (Numeric) → 乘 (Multiply)。
- 添加一个比较函数 (大于等于): 编程 (Programming) → 比较 (Comparison) → 大于等于? (Greater or Equal?)。
- 添加 5 个数值常量: 编程 (Programming) → 数值 (Numeric) → 数值常量 (Numeric Constant), 值分别为 4、6、1、50、30。
- 添加一个条件结构 1: 编程 (Programming) → 结构 (Structures) → 条件结构 (Case Structure)。
- 将 VISA 资源名称 (VISA resource name) 函数的输出端口与串口字节数 (VISA Bytes at Serial Port) 函数 (顺序结构 Frame1 中) 的输入端口引用 (reference) 相连。
- 将串口字节数 (VISA Bytes at Serial Port) 函数的输出端口 VISA 资源名称 (VISA resource name) 与串口读取 (VISA Read) 函数的输入端口 VISA 资源名称 (VISA resource name) 相连。
- 将串口字节数 (VISA Bytes at Serial Port) 函数的输出端口 Number of bytes at Serial port 与串口读取 (VISA Read) 函数的输入端口字节总数 (byte count) 相连。
- 将串口读取 (VISA Read) 函数的输出端口读取缓冲区 (read buffer) 与截取字符串 (String Subset) 函数的输入端口字符串 (string) 相连。
- 将数值常量 (值为 4) 与截取字符串 (String Subset) 函数的输入端口偏移量 (offset) 相连。
- 将数值常量 (值为 6) 与截取字符串 (String Subset) 函数的输入端口长度 (length) 相连。
- 将截取字符串 (String Subset) 函数的输出端口子字符串 (substring) 与分数/指数字符串至数值转换 (Fract/Exp String To Number) 函数的输入端口字符串 (string) 相连。
- 将分数/指数字符串至数值转换 (Fract/Exp String To Number) 函数的输出端口数字 (number) 与减号 (Subtract) 函数的输入端口 x 相连。

其他函数添加及连线略。连接好的框图程序如图 13-70 所示。

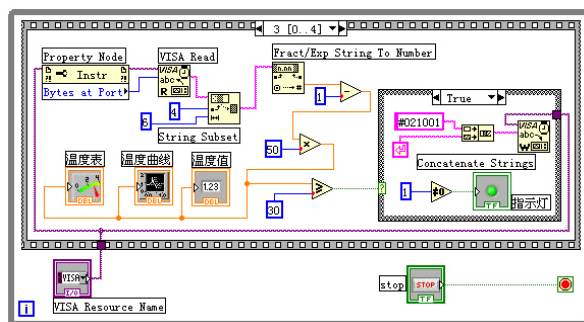


图 13-70 添加接收信息函数

(8) 添加函数到条件结构的真 (True) 选项中。

- 添加一个串口写入函数: 编程 (Programming) → 仪器 I/O (Instrument I/O) → 串口 (Serial) → VISA 写入 (VISA Write)。
- 添加一个字符串常量: 编程 (Programming) → 字符串 (String) → 字符串常量 (String Constant), 值改为 “#021001” (将地址为 2 的模块的开关量输出 0 通道置为高电平)。
- 添加一个回车键常量: 编程 (Programming) → 字符串 (String) → 回车键常量 (Carriage Return)。
- 添加一个字符串连接函数: 编程 (Programming) → 字符串 (String) → 连接字符串 (Concatenate Strings), 用于将读指令和回车符送给写串口函数。
- 添加一个数值常量 (值为 1): 编程 (Programming) → 数值 (Numeric) → 数值常量 (Numeric Constant)。
- 添加一个比较函数不等于 0? (Not Equal To 0?): 编程 (Programming) → 比较 (Comparison) → 不等于 0? (Not Equal To 0?)。
- 将字符串常量 (值为 “#021001”) 与连接字符串 (Concatenate Strings) 函数的输入端口子字符串 (Substring) 相连。
- 将回车键常量与连接字符串 (Concatenate Strings) 函数的另一个输入端口子字符串 (Substring) 相连。
- 将连接字符串 (Concatenate Strings) 函数的输出端口连接的字符串 (Concatenated String) 与 VISA 写入 (VISA Write) 函数的输入端口写入缓冲区 (write buffer) 相连。
- 将数值常量 (值为 1) 与不等于 0? (Not Equal To 0?) 函数的输入端口 x 相连。
- 将指示灯控件图标拖入条件结构的真 (True) 选项中, 将不等于 0? (Not Equal To 0?) 函数的输出端口 x != 0? 与控制指示灯图标相连。

连接好的框图程序如图 13-70 所示。

(9) 添加函数到条件结构的假 (False) 选项中。

- 添加一个串口写入函数: 编程 (Programming) → 仪器 I/O (Instrument I/O) → 串口 (Serial) → VISA 写入 (VISA Write)。
- 添加一个字符串常量: 编程 (Programming) → 字符串 (String) → 字符串常量 (String Constant), 值改为 “#021000” (将地址为 2 的模块的开关量输出 0 通道置为低电平)。
- 添加一个回车键常量: 编程 (Programming) → 字符串 (String) → 回车键常量 (Carriage Return)。
- 添加一个字符串连接函数: 编程 (Programming) → 字符串 (String) → 连接字符串 (Concatenate Strings), 用于将读指令和回车符送给写串口函数。

- 添加一个数值常量 (值为 0): 编程 (Programming) → 数值 (Numeric) → 数值常量 (Numeric Constant)。
- 添加一个比较函数不等于 0? (Not Equal To 0?): 编程 (Programming) → 比较 (Comparison) → 不等于 0? (Not Equal To 0?)。
- 在条件结构的假 (False) 选项中, 添加一个局部变量: 编程 (Programming) → 结构 (Structures) → 局部变量 (Local Variable); 选择局部变量, 单击鼠标右键, 在弹出的快捷菜单的选项 (Select Item) 下, 为局部变量选择对象: “指示灯”, 选择写属性。
- 将字符串常量 (值为 “#021000”) 与连接字符串 (Concatenate Strings) 函数的输入端口子字符串 (Substring) 相连。
- 将回车键常量与连接字符串 (Concatenate Strings) 函数的另一个输入端口子字符串 (Substring) 相连。
- 将连接字符串 (Concatenate Strings) 函数的输出端口连接的字符串 (Concatenated String) 与 VISA 写入 (VISA Write) 函数的输入端口写入缓冲区 (write buffer) 相连。
- 将数值常量 (值为 0) 与不等于 0? (Not Equal To 0?) 函数的输入端口 x 相连。
- 将不等于 0? (Not Equal To 0?) 函数的输出端口 $x \neq 0$ 与局部变量 “指示灯” 相连。
- 将 VISA 资源名称 (VISA resource name) 函数的输出端口分别与条件结构中真 (True) 选项和假 (False) 选项中的串口写入 (VISA Write) 函数的输入端口 VISA 资源名称 (VISA resource name) 相连。

连接好的框图程序如图 13-71 所示。

(10) 在顺序结构 Frame 4 中添加一个时间延迟函数: 编程 (Programming) → 定时 (Timing) → 时间延迟 (Time Delay), 时间采用默认值, 如图 13-72 所示。

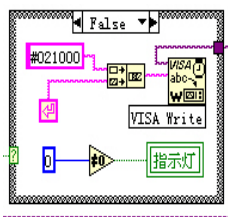


图 13-71 接收信息框图连线

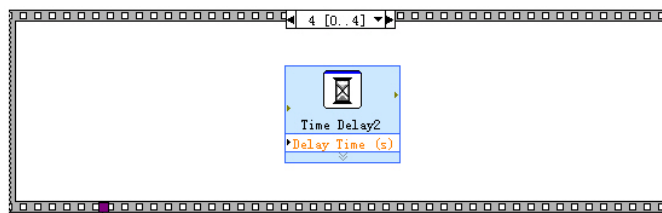


图 13-72 延时框图连线

4. 运行程序

单击快捷工具栏中的“运行 (Run)”按钮, 运行程序。

给传感器升温或降温, 画面中显示测量温度值及实时变化曲线。当测量温度值大于等于 30℃ 时, 画面中指示灯改变颜色, 线路中指示灯亮。

程序运行界面如图 13-73 所示。

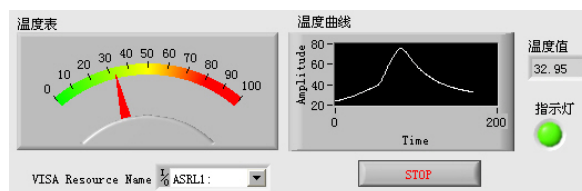


图 13-73 程序运行界面