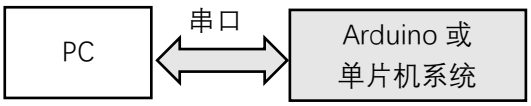
计算机控制与接口课程作业二

姓名：倪怡涛
学号：201820501030

题目：
搭建以下实验系统：

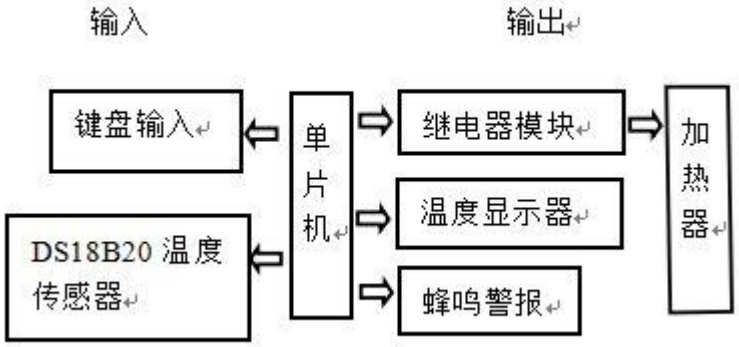


在 PC 中编写软件，运行一仿真程序，模拟一系统（例如温度控制系统，电机控制系统等），要求包含该系统的模型，以及控制接口。该控制接口能够接收来自串口的控制指令。

在 Arduino 中编写控制程序，实现离散 PID 控制。要求该程序包括 PID 控制算法以及控制接口实现，该控制接口能够控制 PC 里的模型程序。

分别运行上述实验系统，在 PC 端记录控制系统的状态曲线，绘制该曲线并机进行说明。

设计：
1.1 总体内容：
利用单片机作为系统的主控制器，利用 DS18B20 作为温度传感器，将信号送入单片机进行处理，经过 PID 算法后，单片机的输出用来控制加热棒的输出功率，从而实现对温度的控制。



2.1 单片机
采用 AT89C52 单片机，其内部有 4KB 单元的程序存储器，不需外部扩展程序存储器，有足够的 io 口。
2.2 温度采集
使用温度传感器 DS18B20，它是最新推出的一种智能型温度传感器，它的优点是可以直接读出被测的温度。主要是对温度信号进行采集和转换工作，电路由 DS18B20 温度传感器和单片机部分组成。温度传感器 DS18B20 把收集到的温度送到单片机的 P2.6 口，单片机接受温度，然后存储下来。

2.3 温度控制

利用单片机控制双向可控硅的导通角。在不同时刻利用单片机给双向可控硅的控制端发出触发信号，使其导通或关断，实现负载电压有效值的不同，以达到调压控制的目的。具体如下：

（1）由硬件完成过零触发环节，即在工频电压下，每 10ms 进行一次过零触发信号，由此信号来达到与单片机的同步。

（2）过零检测信号接至单片机输入口，由单片机对此口进行循环检测，然后进行延时触发。

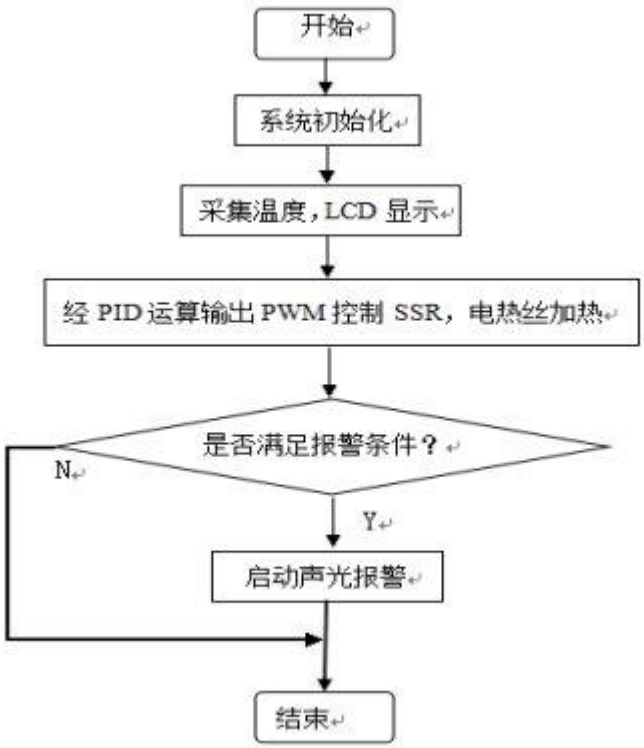
通过单片机控制双向可控硅的导通，从而可以控制加热丝的加热功率。双向可控硅接通，则加热丝加热；双向可控硅断开，则加热丝停止加热。

2.4 液晶显示

LCD1602 液晶显示器显示。

3.总体设计

按照系统设计功能的要求，来确定本系统程序包括 DS18B20 读温度程序、LCD1602 的显示程序、键盘扫描程序、报警处理程序以及继电器加热程序。



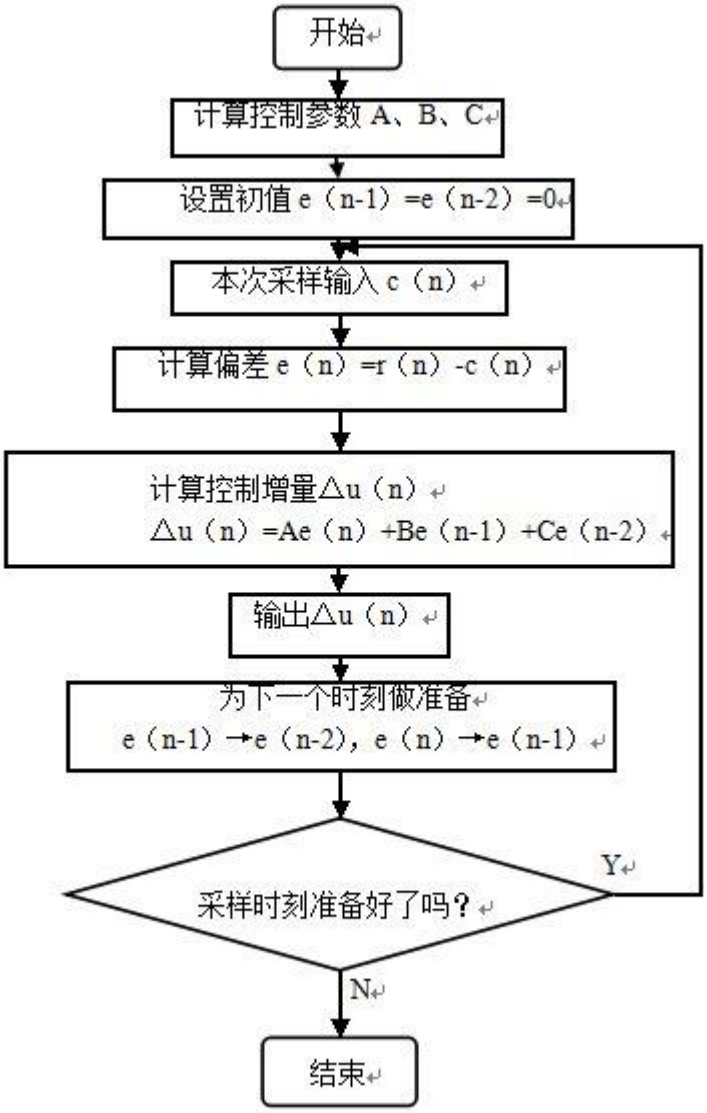
3.1Pid 控制算法：

1 比例环节：把调节器的输入偏差乘以一个系数，作为调节器的输出，比例环节的作用是放大误差的幅值。当仅有比例控制环节时系统输出存在稳态误差。

2 积分环节：调节器加入积分环节后达到消除稳态误差的目的。积分项随着时间的增加而增大，积分项对稳态误差的消除取决于时间的积分。PI 调节器可以使系统在进入稳态后无稳态误差。

3 微分环节：调节器的输入、输出误差信号的微分成正比关系。微分环节能反应偏差信号的变化趋势，从而可以实现超前调节。 控制器中仅有比例环节是不行的，还需要增加微分项，因为它能预测误差变化的趋势。

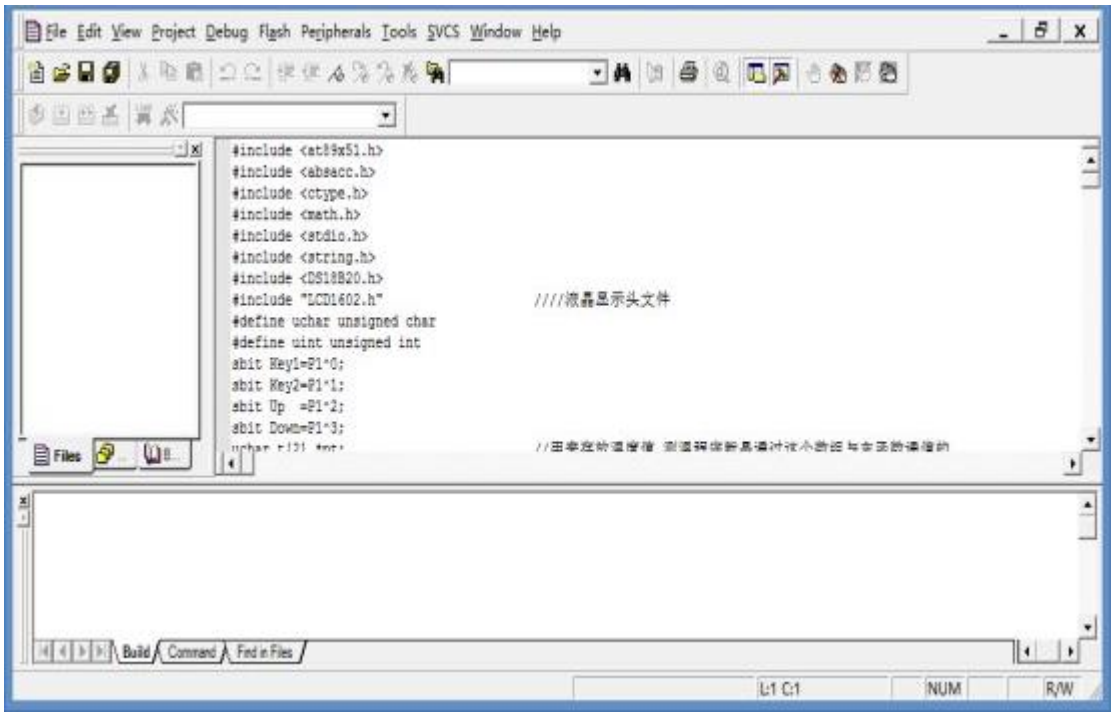
模拟形式	离散化形式
$e(t)=r(t)-c(t)$	$e(n)=r(n)-c(n)$
$\frac{de(t)}{dT}$	$\frac{e(n)-e(n-1)}{T}$
$\int_0^t e(t)dt$	$\sum_{i=0}^{n} e(i)T=T\sum_{i=0}^{n} e(i)$



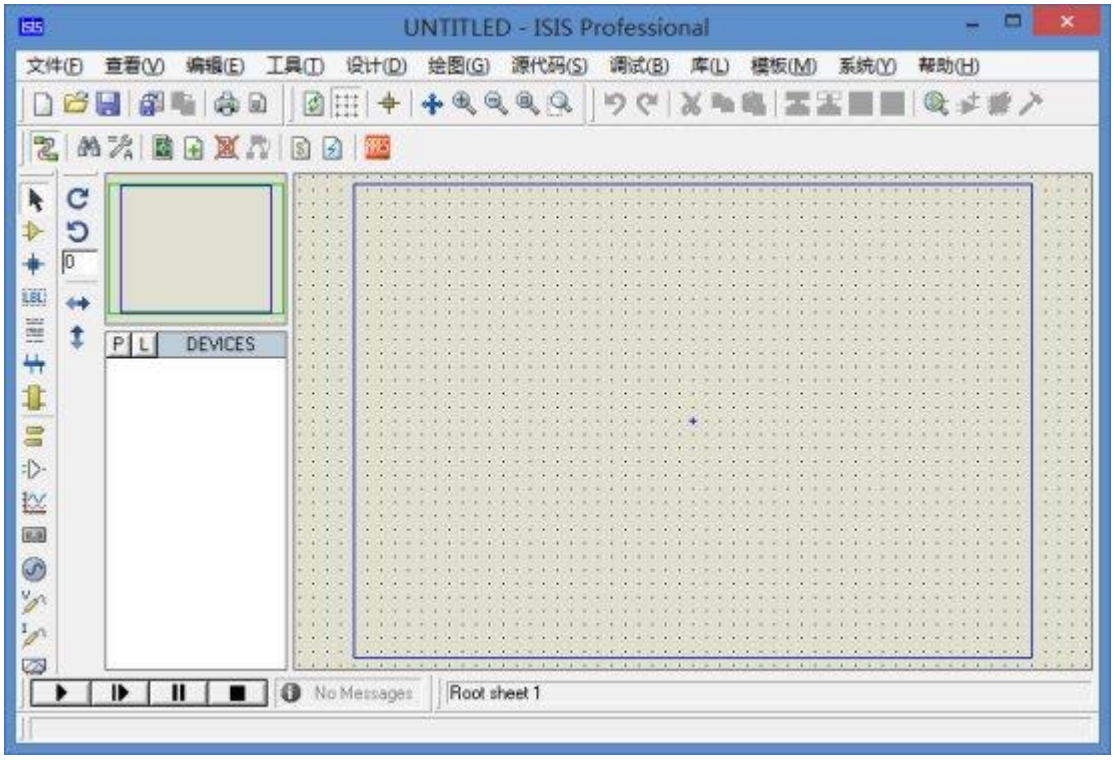
4 系统仿真

对于 AT89C52 的控制设计，编程用 Keil 软件，仿真用 proteus 软件。

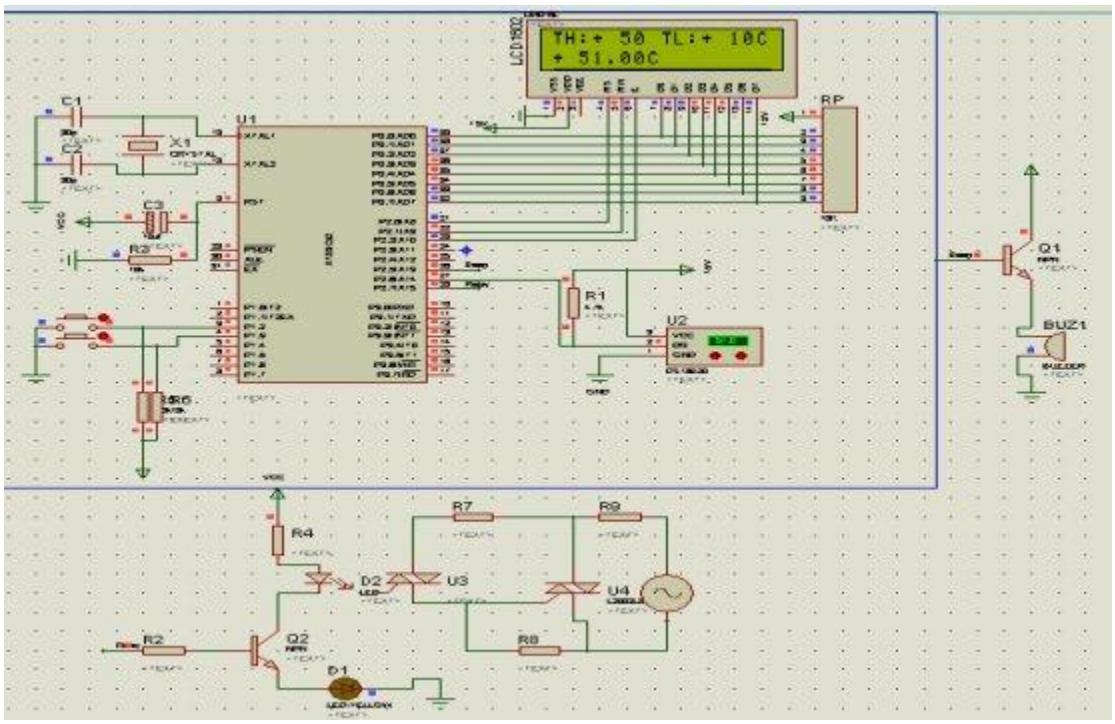
Keil 软件编程如图：



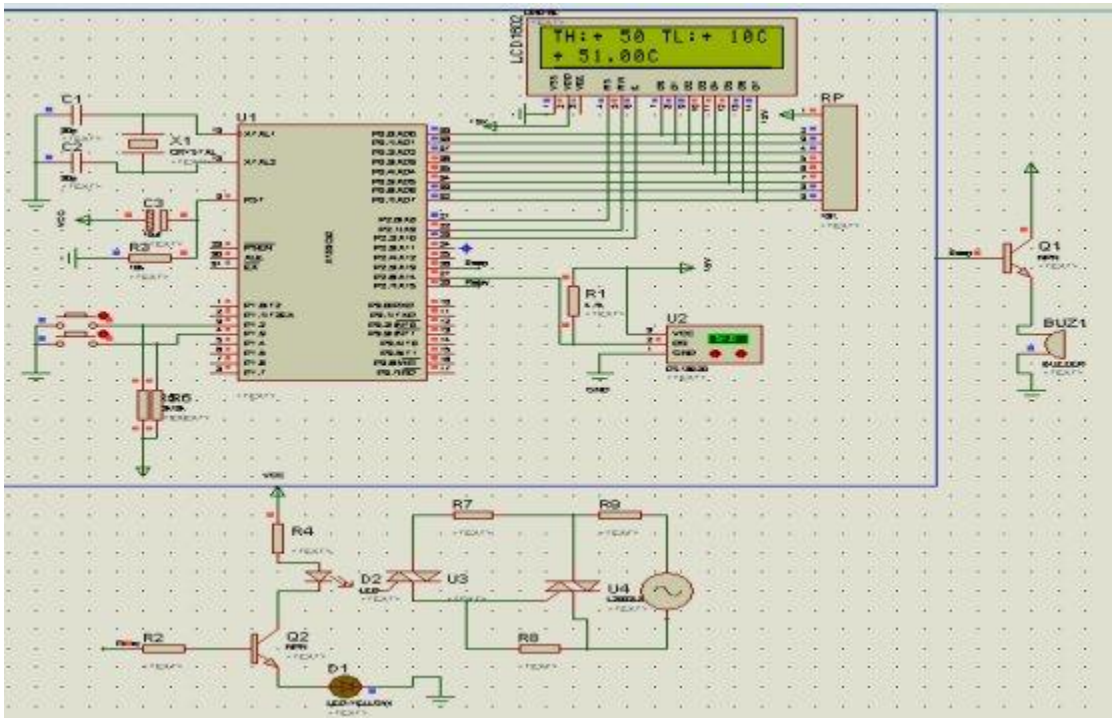
Proteus 软件界面仿真：



当温度还未达到设置温度上限默认的 50℃时，如图所示单片机 P2.5 口输出低电平，蜂鸣器不响。单片机 P2.7 口也输出低电平，二极管不亮，继电器继续加热。

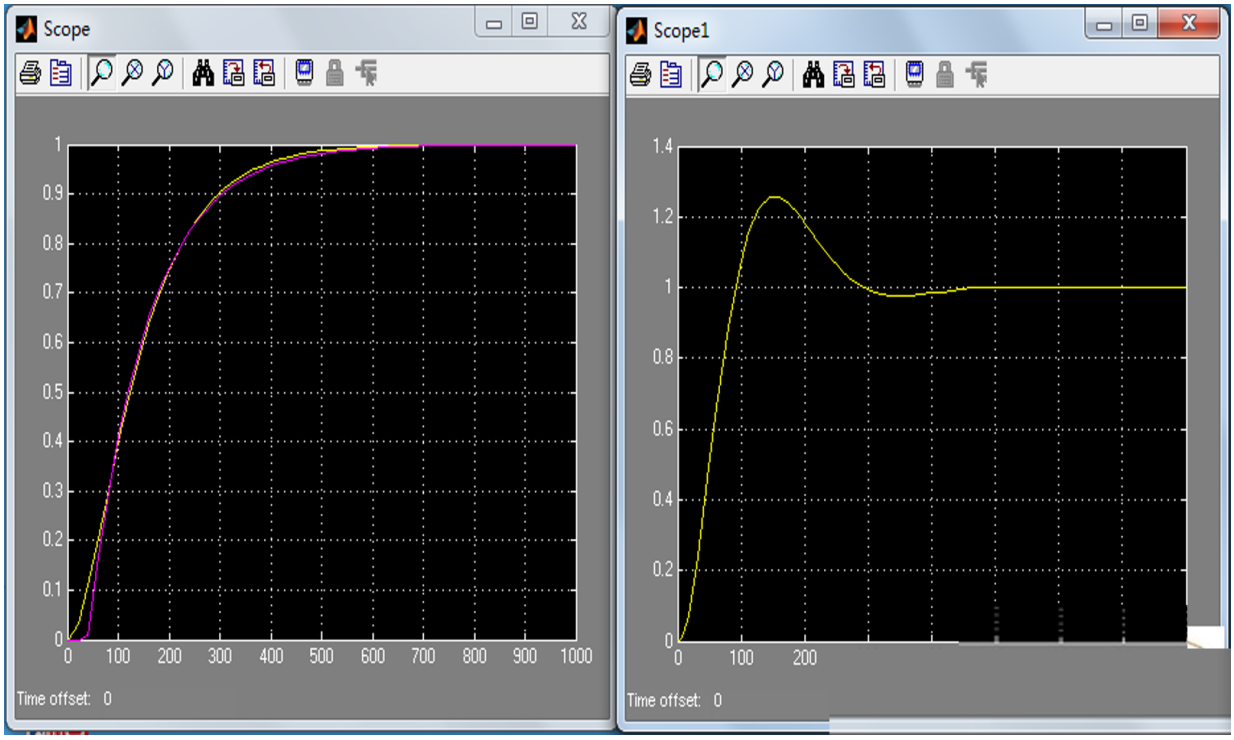


当温度达到设置温度上限默认的 50℃时，如图所示单片机 P2.5 口输出高电平，蜂鸣器响。单片机 P2.7 口也输出高电平，二极管亮，继电器停止加热。



4.2 温度曲线

图中左半部分是系统响应曲线，右半部分是控制效果。


Scope
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0 100 200 300 400 500 600 700 800 900 1000
Time offset: 0
Scope1
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0 100 200
Time offset: 0

程序

```
#include <at89x51.h>
#include <absacc.h>
#include <ctype.h>
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <DS18B20.h>
#include "LCD1602.h"
////液晶显示头文件
#define uchar unsigned char
#define uint unsigned int
sbit Key1=P1^0;
sbit Key2=P1^1;
sbit Up   =P1^2;
sbit Down=P1^3;
uchar t[2],*pt;
//这是用来存放温度值的,测温程序通过这个数组与主函数通信
uchar
TempBuffer1[9]={0x2b,0x31,0x32,0x32,0x2e,0x30,0x30,0x43,'\0'};

//显示实时的温度,上电时显示+125.00C
uchar
TempBuffer0[17]={0x54,0x48,0x3a,0x2b,0x31,0x32,0x35,0x20,

0x54,0x4c,0x3a,0x2b,0x31,0x32,0x34,0x43,'\0'};

//显示温度上下限,上电时显示 TH:+125 TL:+124C
uchar code dotcode[4]={0,25,50,75};
uchar set;//温度初始值
uchar count,high_time=0; //调节占空比的参数
uint rout;// PID 输出
/查表法*******
将表值分离出的十位和个位送到十分位和百分位*********************/

struct PID
{
                uint SetPoint;          // 设定目标 Desired Value
                uint Proportion;        // 比例常数 Proportional Const
                uint Integral;          // 积分常数 Integral Const
                uint Derivative;        // 微分常数 Derivative Const
                signed int LastError;
// 错误 Error[-1]
                signed int PrevError;
// 错误 Error[-2]
                signed int SumError;
// Sums of Errors
};
struct PID spid;                        // PID 控制结构体

void init_pid()
//PID 初始化
{
                high_time=50;
                spid.Proportion = 23;
// Set PID Coefficients
                spid.Integral = 2;
                spid.Derivative =6;
                spid.SetPoint = set;
// Set PID Setpoint
}

unsigned int PIDCalc( struct PID *pp,
unsigned int NextPoint )
//PID 算法
{
                signed int dError,Error;
                Error = pp->SetPoint -
NextPoint;              // 偏差
                pp->SumError += Error;
// 积分
                dError = pp->LastError -
pp->PrevError;      // 当前微分
                pp->PrevError =
pp->LastError;
```

```
                pp->LastError = Error;
                return (pp->Proportion *
Error+ pp->Integral * pp->SumError
+ pp->Derivative * dError);
}

void duty_cycle(uint t)
//  占空比
{
                uchar s;
                t=t/10;
                s=set;
                if(s>t)
                {
                                if(s-t>2)

high_time=100;
                                else
                                {

rout = PIDCalc ( &spid,t );     // Perform
PID Interation

if(high_time<=100)

high_time=(uchar)(rout/600);

else

high_time=100;
                                }
                }
                else

high_time=0;
}

void covert0( unsigned char TH, unsigned
char TL)                    //将温度上下限转
换为 LCD 显示的数据
{
                if(TH>0x7F)
//判断正负,如果为负温,将其转化为其绝对
值
                {

TempBuffer0[3]=0x2d;
//0x2d 为"-"的 ASCII 码
                                TH=~TH;
                                TH++;
                }
                else
TempBuffer0[3]=0x2b;
//0x2B 为"+"的 ASCII 码

                if(TL>0x7f)
                {

TempBuffer0[11]=0x2d;
//0x2d 为"-"的 ASCII 码

TL=~TL+1;
                }
                else

TempBuffer0[11]=0x2b;
//0x2B 为"+"的 ASCII 码


TempBuffer0[4]=TH/100+0x30;
//分离出 TH 的百十个位

if( TempBuffer0[4]==0x30)

TempBuffer0[4]=0xfe; //百位数消隐

TempBuffer0[5]=(TH%100)/10+0x30;
//分离出十位

TempBuffer0[6]=(TH%100)%10+0x30;
//分离出个位

TempBuffer0[12]=TL/100+0x30;
//分离出 TL 的百十个位

if( TempBuffer0[12]==0x30)

TempBuffer0[12]=0xfe; //百位数消隐
```

```
TempBuffer0[13]=(TL%100)/10+0x30;
//分离出十位

TempBuffer0[14]=(TL%100)%10+0x30;
//分离出个位
}


void covert1(void)                    //将温度
转换为 LCD 显示的数据
{
                unsigned char
x=0x00,y=0x00;
                t[0]=*pt;
                pt++;
                t[1]=*pt;
                if(t[1]>0x07)
//判断正负温度
                {

TempBuffer1[0]=0x2d;
//0x2d 为"-"的 ASCII 码

t[1]=~t[1];
/*下面几句把负数的补码*/

t[0]=~t[0];
/*  换算成绝对值*********/
                                x=t[0]+1;
/***********************/
                                t[0]=x;
/***********************/
                                if(x>255)
/***********************/

t[1]++;
/***********************/
                }
                else

TempBuffer1[0]=0x2b;
//0xfe 为变"+"的 ASCII 码
                t[1]<<=4;
//将高字节左移 4 位
                t[1]=t[1]&0x70;
//取出高字节的 3 个有效数字位
                x=t[0];
//将 t[0]暂存到 X,因为取小数部分还要用
到它
                x>>=4;
//右移 4 位
                x=x&0x0f;
//和前面两句就是取出 t[0]的高四位
                t[1]=t[1]|x;
//将高低字节的有效值的整数部分拼成一
个字节

TempBuffer1[1]=t[1]/100+0x30;
//+0x30  为变  0~9 ASCII 码

if( TempBuffer1[1]==0x30)

TempBuffer1[1]=0xfe; //百位数消隐

TempBuffer1[2]=(t[1]%100)/10+0x30;
//分离出十位

TempBuffer1[3]=(t[1]%100)%10+0x30;
//分离出个位
                t[0]=t[0]&0x0c;
//取有效的两位小数
                t[0]>>=2;
//左移两位,以便查表
                x=t[0];
                y=dotcode[x];
//查表换算成实际的小数

TempBuffer1[5]=y/10+0x30;
//分离出十分位

TempBuffer1[6]=y%10+0x30;
//分离出百分位
}

void delay(unsigned int i)
{
```

```
                while(i--);
}

void t0_int(void) interrupt 1
//PWM 波输出
{

if(++count<=(high_time))
                Relay=0;
                else if(count<=100)
                Relay=1;
                else
                count=0;
                TH0=0X20;
                TL0=0X00;
}

main()
{
                unsigned char
TH=50,TL=10,temp=0;
//下一步扩展时可能通过这两个变量,调节
上下限

//测温函数返回这个数组的头地址
                TMOD=0X01;
                TH0=0X20;
                TL0=0X00;
                EA=1;
                ET0=1;
                TR0=1;
                set=60;//目标温度
                Beep=0;Relay=0;
                init_pid();
                while(1)
                {
                    if(Up==0)
                                {

while(!Up);
                                    TH++;
                                }

if(Down==0)
                                {

while(!Down);

TL--;
                                }

pt=ReadTemperature(TH,TL,0x3f);
//上限温度-22,下限-24,分辨率 10 位,也就
是 0.25C

//读取温度,温度值存放在一个两个字节的
数组


if((t[1]>TH)|(t[1]<TL))
                                {

Beep=1;
                        //
Relay=1;

                                }
                                else
                                {

Beep=0;
                        //
Relay=0;
                                }

delay(10000);
                                covert1();

covert0(TH,TL);

LCD_Initial();
//第一个参数列号,第二个为行号,为 0 表示
第一行

//为 1 表示第二行,第三个参数为显示数据
的首地址

LCD_Print(0,0,TempBuffer0);
```

```
LCD_Print(0,1,TempBuffer1);
                }
}
```