|  |  |  |  |
| --- | --- | --- | --- |
| **课程名称** | 微处理器系统结构与嵌入式系统设计（1） | **课程编号** | A2010451 |
| **实验地点** | 电子系统综合设计实验室1YF101 | **实验时间** | 第十六周第5、6、7、8节 |
| **校外指导教师** |  | **校内指导教师** | 夏绪玖 |
| **实验名称** | 基于DS18B20的直流电机转速控制系统 | | |
| **评阅人签字** |  | **成绩** |  |
| 一、实验目的  通过DS18B20温度传感器对实时环境温度进行测量，通过温度的大小对直流电机的转速进行控制，同时在LCD1602上显示当前转速对应的档位，实现自动控制转速。  二、实验原理  2.1 DS18B20温度传感器原理  2.1.1 DS18B20介绍  DS18B20数字温度计提供 9 位温度读数，指示器件的温度。信息经过单线接口送入DS18B20 或从DS1820送出，因此从中央处理器到 DS1820 仅需连接一条线（和地）。读、写和完成温度变换所需的电源可以由数据线本身提供，而不需要外部电源。  2.1.2 DS18B20的工作方式  （1）DS18B20的初始化：主机首先发出一个480－960微秒的低电平脉冲，然后释放总线变为高电平，并在随后的480微秒时间内对总线进行检测，如果有低电平出现说明总线上有器件已做出应答。若无低电平出现一直都是高电平说明总线上无器件应答。作为从器件的DS18B20在一上电后就一直在检测总线上是否有480－960微秒的低电平出现，如果有，在总线转为高电平后等待15－60微秒后将总线电平拉低60－240微秒做出响应存在脉冲，告诉主机本器件已做好准备。若没有检测到就一直在检测等待。  1.jpg  图1 DS18B20的初始化时序图  （2）DS18B20的读操作：首先DS18B20将数据线拉低“0”。之后延时1微秒，将数据线拉高“1”,释放总线准备读数据。再延时10微秒，此时读数据线的状态得到1个状态位，并进行数据处理。最后延时45微妙，重复上述步骤，直到读完一个字节。  3.jpg  图2 DS18B20的读操作时序图  （3）DS18B20的写操作：首先将数据线置低电平“0”，之后延时15微秒，按从低位到高位的顺序发送数据(一次只发送一位)。然后延时60微秒，将数据线拉到高电平。重复上述步骤，直到发送完整的字节，最后将数据线拉高，即可完成一次写操作。  2.jpg  图3 DS18B20的写操作时序图  2.1.3 DS18B20温度转换规则  DS18B20的核心功能是它可以直接读出数字的温度数值。温度传感器的精度为用户可编程的9，10，11或12位，分别以0.5℃，0.25℃，0.125℃和0.0625℃增量递增。在上电状态下默认的精度为12位。  DS18B20启动后保持低功耗等待状态,当需要执行温度测量和AD转换时，总线控制器必须发出[44h]命令。转换完以后，产生的温度数据以两个字节的形式被存储到高速暂存器的温度寄存器中，DS18B20继续保持等待状态。  如图4所示，这是12位转化后得到的12位数据，存储在DS18B20的两个8位的RAM中，高字节的前5位是符号位，如果测得的温度大于0，这5位为“0”，只要将测到的数值乘以0.0625即可得到实际温度；如果温度小于0，这5位为“1”，测到的数值需要先减1再取反再乘以0.0625即可得到实际温度。    图4 DS18B20温度寄存器格式  2.2 直流电机驱动原理  2.2.1 直流电机介绍  直流电机是一种将电能转换为机械能的装置。一般的直流电机有两个电极, 主要由永磁体(定子)、线圈(转子)和换向器组成。当电极正接时，电机正转，当电极反接时，电机反转。直流电机除直流电机外，常见的电机还有步进电机、舵机、无刷电机、空心杯电机等。  2.2.2 直流电机调速原理  对直流电机调速可以采用多种办法。对于直流电机转速的调节，最常用的办法是通过改变电枢端的电压来实现。其中PWM（Pulse Width Modulation）控制是常用的一种调速方法。PWM控制是指在保持周期不变的情况下，通过调节开关导通的时间对脉冲宽度进行调制，从而达到调节电机转速的目的。  在脉宽调速系统中，电机电枢两端的电压是脉宽可调的脉冲电压，在输出脉冲频率足够快的情况下，由于惯性的存在，只要按照一定的规律改变通、断电的时间，即可使电机的速度达到并保持一个稳定值。  2.2.3 PWM基本原理  在 PWM 驱动控制的调整系统中，按一个固定的频率来接通和断开电源，并根据需要改变一个周期内“接通”和“断开”时间的长短。通过改变直流电机电枢上电压的“占空比”来改变平均电压的大小，从而控制电动机的转速。如图5所示，在脉冲作用下，当电机通电时，速度增加；电机断电时，速度逐渐减少。只要按一定规律，改变通、断电的时间，即可让电机转速得到控制。    图5 PWM基本原理  设电机始终接通电源时，电机转速最大为，占空比为，则电机的平均速度为    由上式可知，当我们改变占空比时，可以得到不同的电机平均速度，从而达到调速的目的。严格地讲，与并不是严格的线性关系，在一般的应用中，可以近似的看成线性关系。  2.3 I2C总线的串行扩展  2.3.1 I2C串行总线系统的基本结构  I2C总线只有两条信号线，一条是数据线SDA，另一条是时钟线SCL。SDA和SCL是双向的， I2C总线上各器件的数据线都接到SDA线上，各器件时钟线均接到SCL线上，系统基本结构如图6所示。  IIIICCCCC  图6 I2C串行总线系统的基本结构  2.3.2 I2C串行总线系统的数据传输格式  I2C总线上每传送一位数据都与一个时钟脉冲相对应，传送的每一帧数据均为一字节。但启动I2C总线后传送的字节数无限制，只要求每传送一个字节后，对方回答一个应答位。在时钟线为高电平期间，数据线的状态就是要传送的数据。数据线上数据的改变必须在时钟线为低电平期间完成。在数据传输期间，只要时钟线为高电平，数据线都必须稳定，否则数据线上的任何变化都当作起始或终止信号。  I2C总线数传须遵循规定格式。图7所示为一次完整的数据传送应答时序。根据总线规范，起始信号表明一次数传的开始，其后为寻址字节。在寻址字节后是按指定读、写的数据字节与应答位。在数传完成后主器件都必须发送终止信号。在起始与终止信号之间传输的数据字节数由主器件（单片机）决定，理论上讲没有字节限制。  L9  图7 I2C总线一次完整的数据传送应答时序  2.3.3 AT24C02功能介绍  AT24C02是一个2K位串行CMOS—E2PROM，内部含有256个8位字节。CATALYST公司的先进CMOS 技术实质上减少了器件的功耗。AT24C02有一个16字节页写缓冲器，该器件通过I2C总线接口进行操作，有一个专门的写保护功能。  三、使用仪器及软件  Keil uVision5、Proteus 8 Professional、PZ-ISP普中自动下载软件  普中HC6800-ES V2.0 单片机开发实验仪  四、实验方案设计  4.1 程序流程图    图8 直流电机转速控制系统程序流程图  4.2 单片机模拟输出PWM信号方案设计  PWM 信号的产生通常有两种方法: 一种是软件的方法；另一种是硬件的方法。本方案采用软件的方法实现模拟输出PWM信号，即通过AT89C51单片机内置的定时器T0、T1输出占空比不同的脉冲波形。  选取定时器T1为控制核心，设定其工作方式为方式1，定时时间为0.5ms，则定时器应赋初值为TH1 = 0xFE，TL1 = 0x0C。同时设置一个全局变量timer1用于计数，每当T1溢出则timer1加1，表示时间经过0.5ms。  在程序中，设置了两个全局变量PWM\_high和PWM\_low，分别表示高电平、低电平持续的时间，并固定PWM\_low为10，仅改变PWM\_high的值。当timer1的值小于PWM\_high时，将P1.0置1，表示打开直流电机；当timer1的值大于等于PWM\_high的值且小于PWM\_high和PWM\_low的值之和时，将P1.0置0，表示关闭直流电机，由此即可实现模拟输出PWM信号，且占空比为  4.3 DS18B20温控模块方案设计  在程序中，设置了两个全局变量temperature、DTemp[8]，前者用于存储当前温度（浮点型，保留2位小数）；后者是一个8位数组，其中前4位用于存放温度各个位对应的数字，最后一位存放当前直流电机运行的档位。当按键K4未被按下时，Mode为0，表示此时以断电前AT24C02中存储的PWM\_high调整电机转速；当按键K4按下后，Mode为1，表示此时根据环境温度自动调整电机转速。  各档位与环境温度满足以下关系：   |  |  | | --- | --- | | 温度 | 档位 | |  | 0 | |  | 1 | |  | 2 |   每进行一次温度测量，便会根据测量值调整运行档位，同时通过I2C串行总线向AT24C02中的地址1处写入最新的PWM\_high。  五、实验步骤  （1）确定实验要求，绘制程序流程图。  （2）编写程序源代码。  （3）编译生成hex文件，下载至单片机上进行测试。  （4）分析实验结果并记录。  六、实验结果及分析  启动程序  C:/Users/86173/AppData/Local/Temp/picturecompress_20201222220925/output_1.jpgoutput_1  图9 启动单片机  按下K4清除预留数据  IMG_20201222_213942  图10 按下K4  握住温度传感器使温度升高到20度，进入风扇档 1  IMG_20201222_213951  图11 温度升高超过20度  握住温度传感器使温度超过26度，进入风扇档 2  IMG_20201222_214029  图12 温度升高超过26度  重启单片机读取预留数据为风扇档 2  IMG_20201222_214038  图13 重启单片机  程序经硬件仿真，能实现用温度传感器接收温度变化，从而使风扇进行自动换挡的功能。  七、实验体会  通过这次实验让我们学会了使用LCD数码管，内容显示的控制，实现显示温度和电风扇挡位功能。更加深刻地理解了DS18B20温度传感器的应用，可以使用DS18B20作为信息的采集的端口，为电风扇挡位控制提供控制自变量。理解了步进电机驱动的原理，以及通过脉冲频率调节电风扇转速的方法。更让培养了我们了如何在网上查找资料并一步步解决问题的能力。  附录 实验程序  1—main.c  #include <reg52.h>  #include <temp.h>  #include <i2c.h>  #define LED P0  //----------位定义----------//  sbit LED7 = P2 ^ 7;//标志LED，LED亮起表示以断电前AT24C02中存储的档位运行，熄灭表示根据温度调整档位  sbit PWM = P1 ^ 0;//直流电机开关  sbit k4 = P3 ^ 3;//独立按键K4，用于切换模式  sbit LA = P2 ^ 2;  sbit LB = P2 ^ 3;  sbit LC = P2 ^ 4;  //----------全局变量----------//  unsigned char timer1;  int PWM\_high = 0;//高电平持续的时间  int PWM\_low = 10;//低电平持续的时间，10个定时器时间5ms  float temperature = 0;//当前温度  int Mode = 0;//Mode为0：以断电前AT24C02中存储的档位运行  uchar LED\_CODE[12] = { 0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x0,0x40 }; // 0-9、不显示字符、字符"-"对应的编码  uchar SEG\_CODE[8][3] = { {1,1,1}, {1,1,0}, {1,0,1}, {1,0,0}, {0,1,1}, {0,1,0}, {0,0,1}, {0,0,0} }; // 段码  uint DTemp[8] = { 10, 10, 10, 10, 10, 10, 10, 1 }; // LED动态显示的数字，前4位为温度，最后一位表示档位  //----------函数列表----------//  void Int1Init();//外部中断初始化  void Time1Config();//定时器初始化  void delay(uint i);//延时函数  void Motor\_speed();//电机转速函数  void datapros(int temp);//温度处理函数  void temp\_select();//温度控制档位函数  void DigDisplay();//LED数码管动态显示函数  void Set\_Gear();//根据PWM\_high的大小设置档位  //----------Main函数----------//  void main()  {  Time1Config();  Int1Init();  Ds18b20Init();  PWM\_high = At24c02Read(1);//读取EEPROM地址1内的数据保存在PWM\_high中  while (1)  {  Set\_Gear();  DigDisplay();  Motor\_speed();  if (Mode == 0)  //Mode=0表示：转速维持断电前的档位，不根据温度进行调整，此时LED7会亮起  {  LED7 = 0;  }  else  //Mode=1表示：根据温度调整档位，此时LED7会熄灭  {  LED7 = 1;  temp\_select();  }  }  }  void delay(uint i)  {  while (i--);  }  void Time1Config()  {  TMOD = 0x10;  TH1 = 0xFE;//定时0.5ms  TL1 = 0x0C;  ET1 = 1;  EA = 1;  PT1 = 0;  TR1 = 1;  }  void Motor\_speed()  {  if (timer1 > (PWM\_high + PWM\_low))  timer1 = 0;  if (timer1 < PWM\_high)  //如果计时在高电平时间内，则PWM输出高电平  PWM = 1;  else  //如果在低电平时间内，则PWM输出低电平  PWM = 0;  }  void Time1() interrupt 3  {  TH1 = 0xFE;  TL1 = 0x0C;  timer1++;  }  void datapros(int temp)  {  float tp, tp100;  if (temp < 0) //当温度值为负数  {  //因为读取的温度是实际温度的补码，所以减1，再取反求出原码  temp = temp - 1;  temp = ~temp;  tp = temp;  temp = tp \* 0.0625 \* 100 + 0.5;  tp = tp \* 0.0625;  //留两个小数点就\*100，+0.5是四舍五入，因为C语言浮点数转换为整型的时候把小数点  //后面的数自动去掉，不管是否大于0.5，而+0.5之后大于0.5的就是进1了，小于0.5的就  //算加上0.5，还是在小数点后面。  temperature = tp;  }  else  {  tp = temp;//因为数据处理有小数点所以将温度赋给一个浮点型变量  //如果温度是正的那么，那么正数的原码就是补码它本身  temp = tp \* 0.0625 \* 100 + 0.5;  tp = tp \* 0.0625;  //留两个小数点就\*100，+0.5是四舍五入，因为C语言浮点数转换为整型的时候把小数点  //后面的数自动去掉，不管是否大于0.5，而+0.5之后大于0.5的就是进1了，小于0.5的就  //算加上0.5，还是在小数点后面。  temperature = tp;  }  tp100 = temp % 10000 / 1000 \* 1000 + temp % 1000 / 100 \* 100 + temp % 100 / 10 \* 10 + temp % 10;  DTemp[0] = (int)tp100 / 1000;  DTemp[1] = (int)tp100 % 1000 / 100;  DTemp[2] = (int)tp100 % 100 / 10;  DTemp[3] = (int)tp100 % 10;  }  void temp\_select(void)  {  datapros(Ds18b20ReadTemp());  if (temperature < 20)  {  PWM\_high = 0;  At24c02Write(1, PWM\_high);//在地址1内写入数据PWM\_high  }  if ((temperature < 26) && (temperature >= 20))  {  PWM\_high = 60;  At24c02Write(1, PWM\_high);//在地址1内写入数据PWM\_high  }  if (temperature >= 26)  {  PWM\_high = 52;  At24c02Write(1, PWM\_high);//在地址1内写入数据PWM\_high  }  }  void DigDisplay()  {  uchar c;  for (c = 0; c < 8; c++)  {  LC = SEG\_CODE[c][0]; LB = SEG\_CODE[c][1]; LA = SEG\_CODE[c][2];  if (c != 1)  LED = LED\_CODE[DTemp[c]];  else  LED = LED\_CODE[DTemp[c]] | 0x80;  delay(100);  LED = 0x0;  }  }  void Set\_Gear()  {  switch (PWM\_high)  {  case 0:DTemp[7] = 0; break;  case 60:DTemp[7] = 1; break;  case 52:DTemp[7] = 2; break;  }  }  void Int1Init()  {  IT1 = 1;  EX1 = 1;  PX1 = 1;  EA = 1;  }  void Int1() interrupt 2  {  delay(100);  if (k4 == 0)  Mode = !Mode;  }  2-temp.c  #include"temp.h"  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* 函 数 名 : Delay1ms  \* 函数功能 : 延时函数  \* 输 入 : 无  \* 输 出 : 无  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Delay1ms(uint y)  {  uint x;  for (; y > 0; y--)  {  for (x = 110; x > 0; x--);  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* 函 数 名 : Ds18b20Init  \* 函数功能 : 初始化  \* 输 入 : 无  \* 输 出 : 初始化成功返回1，失败返回0  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  uchar Ds18b20Init()  {  uchar i;  DSPORT = 0; //将总线拉低480us~960us  i = 70;  while (i--);//延时642us  DSPORT = 1; //然后拉高总线，如果DS18B20做出反应会将在15us~60us后总线拉低  i = 0;  while (DSPORT) //等待DS18B20拉低总线  {  Delay1ms(1);  i++;  if (i > 5)//等待>5MS  {  return 0;//初始化失败  }  }  return 1;//初始化成功  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* 函 数 名 : Ds18b20WriteByte  \* 函数功能 : 向18B20写入一个字节  \* 输 入 : 无  \* 输 出 : 无  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Ds18b20WriteByte(uchar dat)  {  uint i, j;  for (j = 0; j < 8; j++)  {  DSPORT = 0; //每写入一位数据之前先把总线拉低1us  i++;  DSPORT = dat & 0x01; //然后写入一个数据，从最低位开始  i = 6;  while (i--); //延时68us，持续时间最少60us  DSPORT = 1; //然后释放总线，至少1us给总线恢复时间才能接着写入第二个数值  dat >>= 1;  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* 函 数 名 : Ds18b20ReadByte  \* 函数功能 : 读取一个字节  \* 输 入 : 无  \* 输 出 : 无  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  uchar Ds18b20ReadByte()  {  uchar byte, bi;  uint i, j;  for (j = 8; j > 0; j--)  {  DSPORT = 0;//先将总线拉低1us  i++;  DSPORT = 1;//然后释放总线  i++;  i++;//延时6us等待数据稳定  bi = DSPORT; //读取数据，从最低位开始读取  /\*将byte左移一位，然后与上右移7位后的bi，注意移动之后移掉那位补0。\*/  byte = (byte >> 1) | (bi << 7);  i = 4; //读取完之后等待48us再接着读取下一个数  while (i--);  }  return byte;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* 函 数 名 : Ds18b20ChangTemp  \* 函数功能 : 让18b20开始转换温度  \* 输 入 : 无  \* 输 出 : 无  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Ds18b20ChangTemp()  {  Ds18b20Init();  Delay1ms(1);  Ds18b20WriteByte(0xcc); //跳过ROM操作命令  Ds18b20WriteByte(0x44); //温度转换命令  //Delay1ms(100); //等待转换成功，而如果你是一直刷着的话，就不用这个延时了  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* 函 数 名 : Ds18b20ReadTempCom  \* 函数功能 : 发送读取温度命令  \* 输 入 : 无  \* 输 出 : 无  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Ds18b20ReadTempCom()  {  Ds18b20Init();  Delay1ms(1);  Ds18b20WriteByte(0xcc); //跳过ROM操作命令  Ds18b20WriteByte(0xbe); //发送读取温度命令  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* 函 数 名 : Ds18b20ReadTemp  \* 函数功能 : 读取温度  \* 输 入 : 无  \* 输 出 : 无  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  int Ds18b20ReadTemp()  {  int temp = 0;  uchar tmh, tml;  Ds18b20ChangTemp(); //先写入转换命令  Ds18b20ReadTempCom(); //然后等待转换完后发送读取温度命令  tml = Ds18b20ReadByte(); //读取温度值共16位，先读低字节  tmh = Ds18b20ReadByte(); //再读高字节  temp = tmh;  temp <<= 8;  temp |= tml;  return temp;  }  3-i2c.c  #include "i2c.h"  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* 函数名 : Delay10us()  \* 函数功能 : 延时10us  \* 输入 : 无  \* 输出 : 无  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Delay10us()  {  unsigned char a, b;  for (b = 1; b > 0; b--)  for (a = 2; a > 0; a--);  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* 函数名 : I2cStart()  \* 函数功能 : 起始信号：在SCL时钟信号在高电平期间SDA信号产生一个下降沿  \* 输入 : 无  \* 输出 : 无  \* 备注 : 起始之后SDA和SCL都为0  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void I2cStart()  {  SDA = 1;  Delay10us();  SCL = 1;  Delay10us();//建立时间是SDA保持时间>4.7us  SDA = 0;  Delay10us();//保持时间是>4us  SCL = 0;  Delay10us();  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* 函数名 : I2cStop()  \* 函数功能 : 终止信号：在SCL时钟信号高电平期间SDA信号产生一个上升沿  \* 输入 : 无  \* 输出 : 无  \* 备注 : 结束之后保持SDA和SCL都为1；表示总线空闲  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void I2cStop()  {  SDA = 0;  Delay10us();  SCL = 1;  Delay10us();//建立时间大于4.7us  SDA = 1;  Delay10us();  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* 函数名 : I2cSendByte(unsigned char dat)  \* 函数功能 : 通过I2C发送一个字节。在SCL时钟信号高电平期间，保持发送信号SDA保持稳定  \* 输入 : num  \* 输出 : 0或1。发送成功返回1，发送失败返回0  \* 备注 : 发送完一个字节SCL=0,SDA=1  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  unsigned char I2cSendByte(unsigned char dat)  {  unsigned char a = 0, b = 0;//最大255，一个机器周期为1us，最大延时255us。  for (a = 0; a < 8; a++)//要发送8位，从最高位开始  {  SDA = dat >> 7; //起始信号之后SCL=0，所以可以直接改变SDA信号  dat = dat << 1;  Delay10us();  SCL = 1;  Delay10us();//建立时间>4.7us  SCL = 0;  Delay10us();//时间大于4us  }  SDA = 1;  Delay10us();  SCL = 1;  while (SDA)//等待应答，也就是等待从设备把SDA拉低  {  b++;  if (b > 200) //如果超过2000us没有应答发送失败，或者为非应答，表示接收结束  {  SCL = 0;  Delay10us();  return 0;  }  }  SCL = 0;  Delay10us();  return 1;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* 函数名 : I2cReadByte()  \* 函数功能 : 使用I2c读取一个字节  \* 输入 : 无  \* 输出 : dat  \* 备注 : 接收完一个字节SCL=0,SDA=1.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  unsigned char I2cReadByte()  {  unsigned char a = 0, dat = 0;  SDA = 1; //起始和发送一个字节之后SCL都是0  Delay10us();  for (a = 0; a < 8; a++)//接收8个字节  {  SCL = 1;  Delay10us();  dat <<= 1;  dat |= SDA;  Delay10us();  SCL = 0;  Delay10us();  }  return dat;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* 函数名 : void At24c02Write(unsigned char addr,unsigned char dat)  \* 函数功能 : 往24c02的一个地址写入一个数据  \* 输入 : 无  \* 输出 : 无  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void At24c02Write(unsigned char addr, unsigned char dat)  {  I2cStart();  I2cSendByte(0xa0);//发送写器件地址  I2cSendByte(addr);//发送要写入内存地址  I2cSendByte(dat); //发送数据  I2cStop();  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* 函数名 : unsigned char At24c02Read(unsigned char addr)  \* 函数功能 : 读取24c02的一个地址的一个数据  \* 输入 : 无  \* 输出 : 无  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  unsigned char At24c02Read(unsigned char addr)  {  unsigned char num;  I2cStart();  I2cSendByte(0xa0); //发送写器件地址  I2cSendByte(addr); //发送要读取的地址  I2cStart();  I2cSendByte(0xa1); //发送读器件地址  num = I2cReadByte(); //读取数据  I2cStop();  return num;  } | | | |