**小组成员：**

1. **DS18B20原理——**

接下来我将围绕测温原理、时序流程、管脚功能、引脚状态四个部分来介绍DS18B20的原理。

一、测温原理

DS18B20它的测温原理可以从内部结构图中清晰地了解，大家可以看图1。首先，它的核心是温度传感器模块，这部分用于直接感知环境温度，并将温度值转换为数字信号，随后传送到内部的高速缓冲存储器中，方便数据的快速存取。

与外部设备的通信是通过一根数据线实现的，这就是它的64位ROM和单线接口模块。这根单线不仅负责数据传输，还可以为DS18B20供电。ROM中存储了每个DS18B20的唯一地址，使得主控设备能够区分并控制多个DS18B20传感器。

为了实现温度监控和报警，DS18B20还配备了存储器和控制器模块。这个模块中包含了两个可设定的温度阈值，即低温触发器（TL）和高温触发器（TH）。用户可以设置温度的上下限，当温度超出设定范围时，传感器可以触发报警。

在数据传输过程中，CRC生成器模块负责生成校验码，确保传输过程中的数据准确无误，从而提高通信的可靠性。

总的来说，当DS18B20感知到温度变化时，它会将温度数据转换为数字信号并存储在缓冲器中，随后通过单线接口传输给主控设备。同时，通过设置TL和TH阈值，可以实现对温度超限的监控报警功能。

二、时序

DS18B20通过一线接口进行通信，这里的时序控制是其工作原理的关键，如图2所示。主要包括初始化阶段、命令发送、数据读取三个过程。通信开始时，单片机发送一个480μs的低电平信号来唤醒传感器，接着释放总线，DS18B20会响应以表示准备好接受命令。单片机可以发送不同的命令，比如读取温度或写寄存器。每个命令都有特定的时序格式，确保数据的正确传输。

在数据读取过程中，DS18B20完成温度测量后，会将温度数据以字节形式通过一线接口发送给单片机。单片机需要在指定的时间内读取这些数据，以免丢失信息。

三、管脚功能

DS18B20有三个主要引脚。首先是GND引脚，它需要连接到系统地；其次是VDD引脚，它提供3.0V到5.5V的电源；最后是DQ引脚，它用于数据传输，既可以作为输入也可以作为输出。

四、工作时引脚状态

在设备工作时，GND引脚始终连接到地，VDD引脚则连接到电源。而DQ引脚在不同状态下会变化，例如在初始化期间为低电平，而在数据传输时则根据发送的命令和数据状态进行高低电平的交替变化。

结合硬件原理图3，我们通常会看到GND引脚连接到地，VDD引脚连接到电源，而DQ引脚则通过一个上拉电阻（通常为4.7kΩ）连接到VDD电源，同时通过微控制器的相应IO口与之相连。上拉电阻的作用是确保DQ引脚在空闲状态下保持高电平，当需要发送数据时，微控制器会将DQ引脚拉低，从而实现数据传输。

1. **硬件原理图介绍——**

在这个实验中，我们使用Proteus仿真软件搭建了一个数字温度计电路，通过DS18B20温度传感器来测量温度，并使用数码管动态显示测量结果，如图4所示。整个电路的核心是AT89C51单片机，它负责读取DS18B20的温度数据，并将数据处理后输出到数码管进行显示。

首先，DS18B20温度传感器用于实时测量环境温度，并通过单线接口与单片机通信。DS18B20的DQ引脚连接到单片机的一个I/O口，通过这一根数据线，单片机可以控制DS18B20的操作，如发送温度读取指令并接收温度数据。该传感器的输出是数字信号，因此无需额外的模数转换，数据传输过程稳定、可靠。

单片机读取到温度数据后，会将温度值以数字形式传递给数码管进行显示。仿真图中可以看到，单片机的多个引脚连接到数码管的不同段（A到G和DP），通过这些引脚的组合状态，单片机可以控制数码管显示特定的数字。数码管采用动态扫描的方式显示温度，这意味着它会快速地切换显示每一位，从而在视觉上形成一个完整的温度数值显示。

在仿真过程中，我们可以观察到数码管会实时显示DS18B20测得的温度值。当温度变化时，数码管上的数值也会随之改变，准确地反映当前的温度。这种方式模拟了实际应用中数字温度计的显示效果，使我们能够直观地看到温度数据的变化。

通过该仿真电路，我们可以深入理解DS18B20的温度测量和数据传输原理，并掌握使用单片机动态显示温度数据的基本方法。以上就是整个电路的原理和现象介绍。

1. **软件部分DS18B20写函数——**

在这个代码段中，write(int dat)函数用于向DS18B20传感器写入一个字节的数据。这个写操作是通过单线协议实现的，该协议要求逐位写入数据，确保每个位的时序正确。

void write(int dat)：定义了一个名为write的函数，它接受一个整数dat作为参数，代表需要写入DS18B20的8位数据。

int i;：声明了一个整型变量i，用于控制循环次数。在后续的循环中，i将从0递增到7，用于写入每一位数据。

for(i=0;i<8;i++)：使用一个for循环来遍历8次，每次写入数据的一个位。因为DS18B20一次只能接收一位数据，因此需要循环8次来写入整个字节的数据。

dq=0;：将数据线dq置低电平（0）。这是DS18B20通信协议要求的一个起始信号，用于开始每一位的数据写入过程。

\_nop\_();\_nop\_();\_nop\_();\_nop\_();：这是几个空操作指令\_nop\_()，用于产生一个短暂的延时，确保数据线保持低电平的时间符合DS18B20的时序要求。通过连续调用\_nop\_()，可以精确控制低电平持续的时间。

dq=dat&0x01;：将数据线dq设置为dat的最低位（LSB）的值。dat&0x01的操作会取出dat的最低位，如果最低位为1，则dq为高电平；如果最低位为0，则dq为低电平。这一步实现了逐位写入数据。

delayus(5);：调用delayus(5)函数，使数据线保持当前状态（高或低）5微秒，确保DS18B20能正确接收当前写入的这一位数据。这是数据写入的关键延时，符合DS18B20的通信协议。

dq=1;：将数据线dq拉高，以结束当前位的写入。这一步准备好数据线，以便进行下一个数据位的写入。

dat=dat>>1;：将数据dat右移一位，以便在下一次循环中可以继续操作下一个位。通过右移操作，原来的次低位（第二位）成为最低位，这样在下一次循环时能够顺利写入下一个位。

总结来说，write函数依次将数据dat的每一位写入到DS18B20传感器，通过逐位写入并遵循正确的时序控制来完成一个字节的数据传输。

1. **软件部分DS18B20读函数——**

在这个代码段中，read()函数用于从DS18B20传感器读取一个字节的数据。DS18B20的单线协议要求逐位读取数据，因此函数通过循环逐位读取，并最终返回8位的完整数据。

int read()：定义了一个名为read的函数，返回类型为int。该函数用于从DS18B20读取一个字节的数据并返回。

int i, temp=0;：声明了两个整型变量。i用于循环控制，temp用于存储读取到的数据，初始值设为0，以便逐位构造完整的字节数据。

for(i=0; i<8; i++)：使用一个for循环来遍历8次，每次从DS18B20读取一位数据。因为DS18B20一次只能传输一位，因此需要循环8次才能读取到一个完整的字节。

dq=0;：将数据线dq置低电平（0），这是DS18B20读取操作的起始信号，用于通知DS18B20准备发送当前位的数据。

dq=1;：紧接着将数据线dq置高电平（1），此时DS18B20会将当前位的数据放到数据线上，等待主控设备读取。

delayus(1);：调用delayus(1)函数产生1微秒的延时，确保DS18B20的数据线状态稳定后再读取。这是数据读取的关键时序控制，以保证数据正确读取。

temp=temp>>1;：将temp右移一位，为当前读取到的数据位腾出位置。因为数据是从最低位开始读取的，所以每次循环都需要将已有数据右移一位，以确保新读取的位位于正确的位置上。

if(dq==1) temp=temp|0x80;：检查数据线dq的电平状态。如果dq为高电平（1），说明当前读取到的位为1，此时通过temp | 0x80将temp的最高位设置为1。如果dq为低电平（0），则当前读取位为0，temp保持不变。这一步实现了逐位构建数据的过程。

delayus(4);：调用delayus(4)函数，产生4微秒的延时，确保每一位读取之间有足够的间隔时间，符合DS18B20的时序要求。

return(temp);：当8位数据全部读取完成后，返回temp的值。temp中此时存储的是完整的一个字节数据，从而实现了读取操作。

总结来说，read()函数通过将数据线的状态逐位读取并组合，成功从DS18B20接收一个字节的数据。每一位数据通过适当的时序控制和位操作，最终组合成完整的数据字节并返回。

1. **软件部分DS18B20读写函数——**

这段代码用于通过 DS18B20 温度传感器读取温度值。首先，程序声明了两个整数变量 temp1 和 temp2，分别用于存储从传感器读取的低八位和高八位的温度数据。

在代码开始时，dq 引脚被设置为高电平，准备与 DS18B20 进行通信。接着调用 initial() 函数进行必要的初始化，以确保传感器可以正常工作。

随后，程序发送 0xCC 命令，这个命令是跳过 ROM 的命令，使得 DS18B20 能够与所有连接的设备进行通信。接着发送 0x44 命令以启动温度转换过程，这样 DS18B20 会开始测量温度。

温度转换完成后，程序再次将 dq 设置为高电平，并调用 initial() 函数，以准备读取温度数据。再次发送 0xCC 命令来选择 DS18B20，并发送 0xBE 命令以指示传感器读取温度值。

程序随后通过调用 read() 函数两次，分别读取温度的低八位和高八位，分别存储在 temp1 和 temp2 中。

接下来，程序根据 temp2 的值来判断温度是正值还是负值。当 temp2 小于 8 时，表示温度为正，程序将 temp1 和 temp2 组合成完整的温度值 n，然后将其除以 16（即 n / 2 / 2 / 2 / 2），相当于将值转换为摄氏度，单位为 0.0625°C，并设置 flag 为 0，表示当前温度为正。

如果 temp2 大于 8，则表示温度为负值，程序同样组合 temp1 和 temp2 得到完整的温度值 n。此时，程序对 n 进行取反，并加 1，以计算补码形式的负温度值，最后也将其除以 16，将其转换为摄氏度，并将 flag 设置为 1，表示当前温度为负。

通过这种方式，代码能够有效读取 DS18B20 传感器的温度数据，并正确处理正负温度情况。这一过程确保了温度值的准确获取，使得后续的温度控制或监测系统可以依赖于该数据进行操作。

1. **软件部分LED显示函数——**

这段代码的功能是通过 LED 数码管动态显示温度值。程序首先声明了一个静态变量 i 和四个整数变量 n1、n2、n3、n4，其中 i 用于控制当前显示的位数，而 n1 到 n4 则用来存储从温度值 n 中提取出的千位、百位、十位和个位数字。

接下来，程序定义了一个数组 num，该数组存储了数字 0 到 9 在数码管上的编码。每个数字对应一个特定的编码值，这些编码在后续的显示中将被用来点亮相应的数码管。

然后，程序通过算术运算将温度值 n 拆分为四个部分：千位（n1）、百位（n2）、十位（n3）和个位（n4）。这些值将用于控制数码管显示的数字。

接下来，程序将数码管的控制引脚 s1、s2、s3 和 s4 初始化为高电平（关闭状态），确保所有数码管在显示之前都处于关闭状态。

在 switch 语句中，程序根据 i 的值选择当前要显示的数字。在每个 case 中，程序首先检查对应的位是否有有效的数字。如果 n1（千位）不为零，则显示千位数字，并将 s1 设置为低电平，以打开千位数码管的显示；类似地，程序检查百位和十位的条件并更新显示。

当温度值为负时，程序还会显示负号，此时 flag 被用作指示器。

最后，当 i 的值达到 5 时，程序将其重置为 1，以循环显示。程序中还包含一个延时函数 delayms(2)，用于控制数码管的刷新频率，确保显示效果流畅。

通过这一过程，代码实现了对温度值的动态显示，使我们能够清晰地看到当前测量的温度数据。

1. **实验结果分析和调试技巧——**

在本次实验中，我们使用 Proteus 软件搭建了一个数字温度计的仿真电路，并通过 Keil 软件编写相应的程序，利用 DS18B20 进行温度测量，最后将测得的温度通过数码管动态显示出来。

通过实验，我们能够成功读取 DS18B20 传感器的温度值，并在数码管上实时显示。根据我们的测试，传感器在 -10℃ 到 85℃ 的范围内，测量结果与实际温度的偏差在 ±0.5℃ 以内，符合传感器的规格说明。该传感器的快速响应和较高的测量精度使得它非常适合用于数字温度计的应用。

在调试过程中，我们发现了一些有效的技巧。首先，确保 DS18B20 的接线正确，特别是一线串行接口的连接，以避免信号传输不良。其次，使用 Proteus 的调试功能可以帮助我们观察信号波形，确认数据读取的正确性。

编写代码时，注意各个命令的顺序和逻辑关系，确保在启动温度转换后，程序等待足够的时间（500 毫秒）以允许传感器完成测量。使用延时函数控制显示频率，可以使得数码管显示更加流畅。

最后，及时检查传感器的工作状态，通过监控数码管的显示值，确认程序是否正常执行。通过反复调试和测试，我们最终成功实现了数字温度计的功能，并对 DS18B20 的使用有了更深入的理解。

本次实验不仅增强了我们对 DS18B20 数字温度计的操作技能，也提升了我们在实际项目中进行调试和故障排除的能力。

1. **软件优化和实验拓展延申——**

在本次数字温度计实验中，我们不仅学习了如何使用 DS18B20 传感器，还认识到软件优化的重要性。关于实验优化部分，我们查找资料后，做了一个拓扑图，如图6所示。

通过优化程序代码，我们可以提高系统的性能和响应速度。例如，我们可以利用中断机制，通过定时器中断定时读取温度，这样可以避免主循环中的阻塞，提高系统的响应速度和效率。此外，还可以将代码模块化，将温度读取和数据显示等功能分成多个函数，这样不仅增加了代码的可读性，也便于后期的维护。

在数据处理与滤波方面，可以使用滤波算法（如移动平均法）来减少温度数据中的噪声和误差，从而提高测量的精度。同时，设定合理的温度范围，进行异常值处理，确保输出数据的可靠性。

在显示温度数据时，我们可以考虑使用动态显示的技术，优化数码管的刷新率，使其在保证清晰可读的同时，减少功耗。通过引入中断机制，能够在温度变化时自动触发更新，而不是定时刷新，从而提高响应能力和系统效率。

对于实验的进一步拓展，我们可以考虑以下几个方向。首先，可以将数字温度计与其他传感器结合，如湿度传感器，制作一个环境监测系统，如图7所示。通过集成多种传感器数据，我们可以实现更全面的气候监测功能，提供更丰富的信息。

其次，我们可以探索无线传输技术，如图8所示。将测得的温度数据通过蓝牙或 Wi-Fi 模块发送到手机或计算机进行实时监控。这将使得用户能够远程访问温度信息，增加系统的灵活性和便利性。

最后，我们还可以将该数字温度计与微控制器结合，开发一个简单的温控系统，实现对加热或制冷设备的智能控制，如图9所示。例如，当温度超过设定值时，系统可以自动启动风扇或空调进行降温，从而实现智能家居环境的管理。

通过以上软件优化和实验拓展，我们不仅能提升数字温度计的功能和性能，还能激发更多的创意应用，进一步增强我们的实践能力和创新思维。这为我们未来的学习和研究提供了广阔的空间。