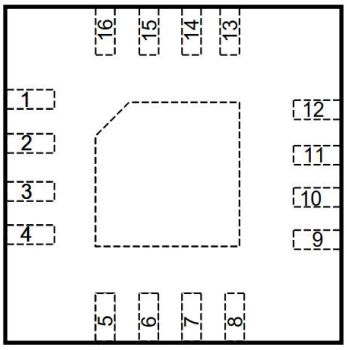
M401 应用指南

一、 内部结构

1.1 管脚定义

QFN16 (3*3mm)封装 ， PIN 2 PIN 兼容替代 TI 的 TMP464



正面透视图

序列	IO 名称	描述	类型	其它
1	NC			
2	NC			
3	RT4	测温通道 4	模拟输入	接 BJT
4	RT3	测温通道 3	模拟输入	接 BJT
5	RT2	测温通道 2	模拟输入	接 BJT
6	RT1	测温通道 1	模拟输入	接 BJT
7	RTN	测温通道公共端	模拟输入	接 BJT
8	GND	地	地	

9	ADD0	I²C 地址选择 1	数字输入	
10	$\overline{\text{HTA}}$	高温阈值报警	数字输出，OD	实现温度报警中断信号，低电平有效。漏极开路，需要在 VDD 间加上拉电阻。
11	$\overline{\text{LTA}}$	低温阈值报警	数字输出，OD	
12	SDA	I²C 数据	数字双向 IO	
13	SCL	I²C 时钟	数字 IO	
14	VDD	电源	电源	2-5V 兼容
15	ADD1	I²C 地址选择 2	数字输入	
16	NC			

1.2 地址配置

M401 可作为 I²C 总线上的从设备与上位机通讯。接口上的 SDA 和 SCL 管脚集成了尖峰抑制滤波器和施密特触发器，从而最小化输入峰值和总线噪声的影响。M401 支持快速模式（1 kHz 到 400 kHz），数据传输是 MSB 模式。

I²C 接口的数据端口 SDA 和时钟端口 SCL 分别连接到上位机处理器的对应端口上，并分别通过上拉电阻 Rp 连到 VDD，通过上位机软件来实现各节点芯片的读写控制。根据实际应用， 可以通过管脚 ADD0 和 ADD 1 的值来设定从设备的地址。

从设备地址定义如下：

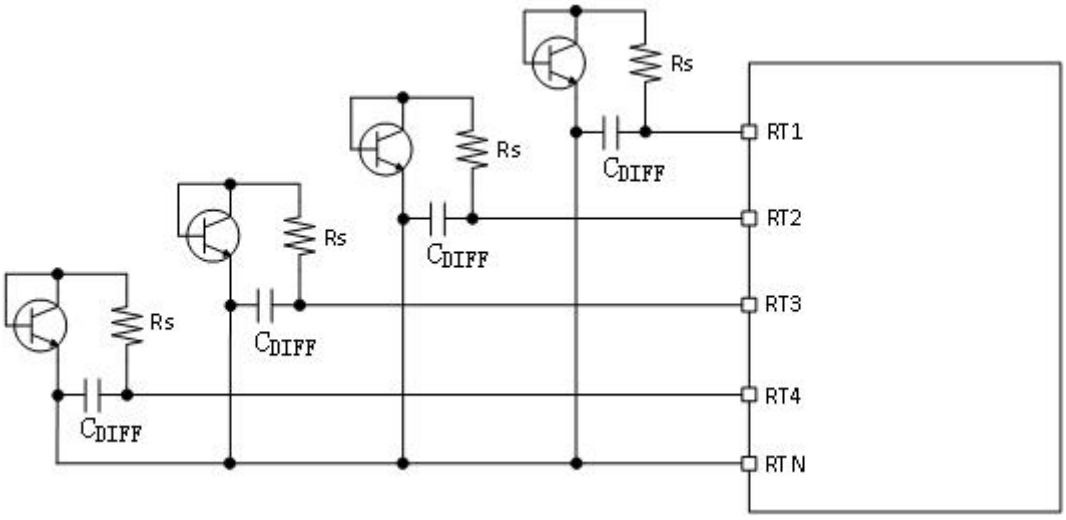
ADD 1	**ADD 0**	**从设备地址**
0	0	0 x 4 8
0	1	0 x 4 9
1	0	0 x 4 A

1	1	0 x 4 B

二、 外围器件参数

(1) 远端 BJT 滤波电路取值

下图为推荐电路，其中 Rs 选择 1.78k、1%精度，C_{DIFF} 选择 1nF。



(2) 芯片 VCC 上并联 2.2uF-4.7uF 电容

(3) 三极管型号推荐：长电 S9018

工作温度范围-55~+150℃，SOT23-3 封装，结温 150℃，主要参数如下表：

Parameter	Symbol	Test conditions	Min	Typ	Max	Unit
Emitter cut-off current	I_{EBO}	V_{EB}=3V，I_C=0			0.1	μA
DC current gain	H_{FE}	V_{CE}=5V, I_C=1mA	28		270	
Collector-emitter saturation voltage	$V_{CE(sat)}$	I_C=10mA,I_B=1mA			0.5	V
Base-emitter saturation voltage	$V_{BE(sat)}$	I_C=10mA,I_B=1mA			1.42	V

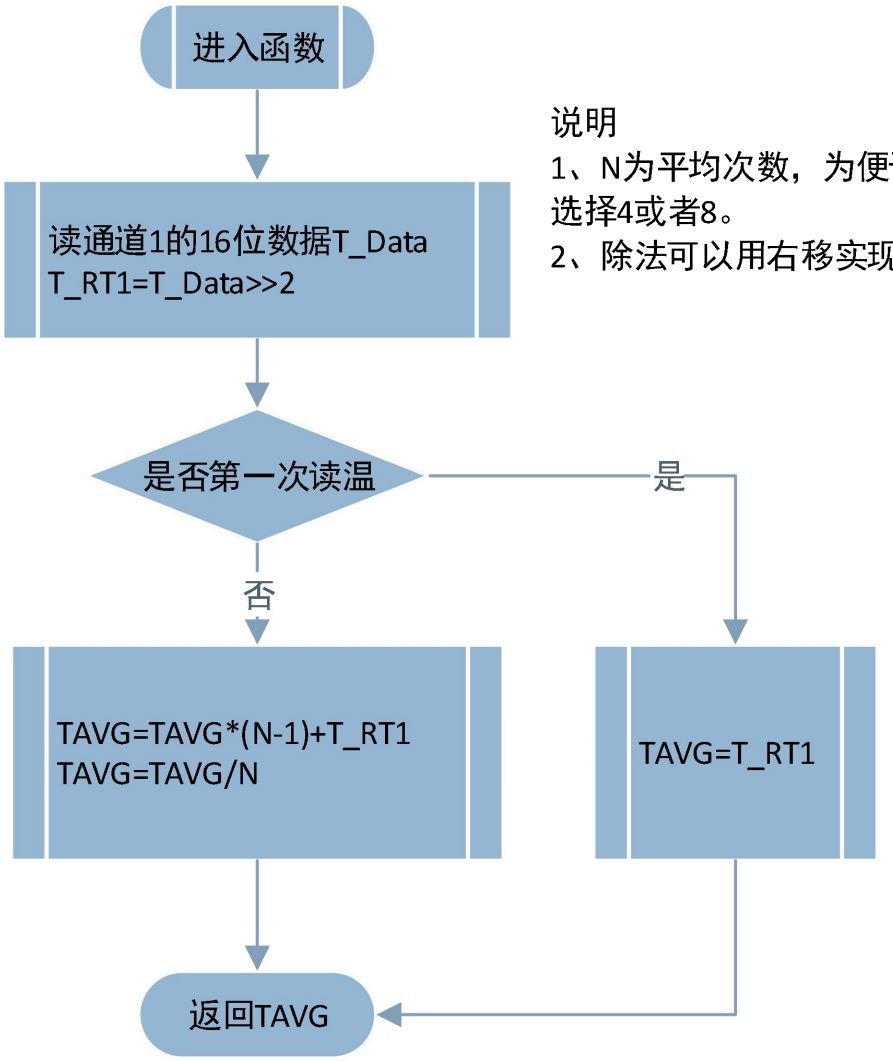
三、 温度数据格式

下表是温度数据格式。每组温度数据占 2 个字节，分为高位字节和低位字节，共 16bit。其中高 14bit 为温度数值，低位字节的 B1 为奇偶校验位，低位字节的 B0 为开路标志位。Local 通道的开路标志一直为 0。

地址	寄存器	默认值	第 7 位	第 6 位	第 5 位	第 4 位	第 3 位	第 2 位	第 1 位	第 0 位
0x0	LTH	0x00	LT13	LT12	LT11	LT10	LT9	LT8	LT7	LT6
0x1	LTL	0x00	LT5	LT4	LT3	LT2	LT1	LT0	PC	0
	本地温度		本地温度值: LT[13:0]； PC: LT[13:0] 偶校验位.							
0x2	RT1H	0x00	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6
0x3	RT1L	0x00	RT5	RT4	RT3	RT10	RT1	RT0	PC	OPEN
	远端温度#1		远端#1 温度: RT[13:0] ； PC: RT[13:0] OPEN=1，远端开路							
0x4	RT2H	0x00	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6
0x5	RT2L	0x00	RT5	RT4	RT3	RT2	RT1	RT0	PC	OPEN
	远程温度#2		远端#2 温度: RT[13:0] ； PC: RT[13:0] OPEN=1，远端开路							
0x6	RT3H	0x00	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6
0x7	RT3L	0x00	RT5	RT4	RT3	RT2	RT1	RT0	PC	OPEN
	远程温度#3		远端#3 温度: RT[13:0] ； PC: RT[13:0] OPEN=1，远端开路							
0x8	RT4H	0x00	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6
0x9	RT4L	0x00	RT5	RT4	RT3	RT2	RT1	RT0	PC	OPEN
	远程温度#4		远端#4 温度: RT[13:0] ； PC: RT[13:0] OPEN=1，远端开路							

四、 温度平均算法

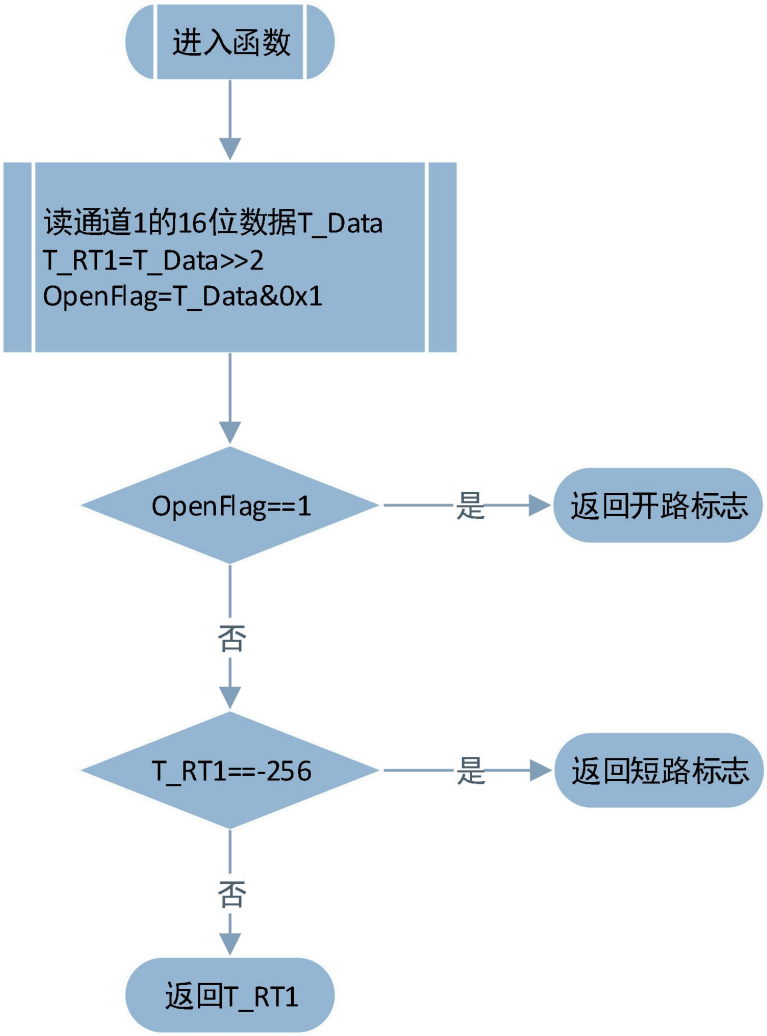
以下算法都以通道 1 为例，算法流程如下图所示。



说明
1、N为平均次数，为便于计算可选择4或者8。
2、除法可以用右移实现。

五、 开短路检测算法

开路短路检测算法如下。



六、 软件控制

6.1 数据偶校验算法

```
//////////////////////////////////////////////////////////////////////

// M401 提供温度测量数据的偶校验（EVEN）功能，偶校验位 PC 位于温度寄存器 LTL 或者 R
TxL 的第 1 位。以 14 位温度测量数据中的 1 的个数作为检错的标志位，若 1 的个数为奇数，则
偶校验位为 1，若 1 的个数为偶数，则偶校验位为 0。

读取通道 x 的 16 位数据 T_Data
```

```
tmp = T_Data >>1;
ecc = tmp&0x1;
cal_ecc = 0;
for(i=0;i<14;i++)
{
        tmp = tmp>>1;
        cal_ecc = cal_ecc^(tmp&0x1);
}
if(cal_ecc==ecc)
{
        printf("The temperature data is correct!\r\n");
}
```

6.2 解除芯片写保护

```
//////////////////////////////////////////////////////////////////////
//// 所有的寄存器都设定为写保护模式以防止软件误操作，写保护模式不影响读取操作。通
过给寄存器（0xE6）写入 0xC4，启动写保护；通过给寄存器（0xE7）写入 0xC4，来解除写
保护。默认上电即为寄存器写保护模式，撤销写保护之后，才能进行寄存器写操作。
void unlock(void)
{
    I2C_WriteOneByte(0xe7, 0xc4);
}
```

6.3 启动芯片写保护

```
//////////////////////////////////////////////////////////////////////
void lock(void)
{
    I2C_WriteOneByte(0xe6, 0xc4);
}
```

6.4 读 1 路本地和 4 路远端温度

```
//////////////////////////////////////////////////////////////////////
```

```
// 主机发送复位脉冲（大于 480us 低电平），并检查存在脉冲
void Read_Temperature(void)
{
    u8 registerValue,channelNumber = 0,i,status;
    s16 data = 0;
    float temp = 0.0f,sum = 0.0f;

    //解除芯片写保护

    unlock();


    //关闭所有测温电路

    I2C_WriteOneByte(0x18, 0x1);


    //循环测温

    while (1)
    {
        if(channelNumber == 0)
        {
            registerValue=0xD;
        }
        else
        {
            registerValue=(0x8<<channelNumber)+0x5;
        }

        //设置温度通道

        I2C_WriteOneByte(0x17, registerValue);

        //每个通道取 4 次温度值，然后取平均，以防止数据抖动

        for (i = 0; i < 4; i++)
        {
            I2C_WriteOneByte(0x18, 0x41);
            delay_ms(25);

            //读取状态寄存器

            status = I2C_ReadOneByte(0x0B);

            //读取状态寄存器的 BUSY 位以确定温度转换是否完成

            if(status&0x80)
            {
```

```
            printf("温度转换未完成！ REG 0x%02x --- Value ---0x%02x \r\
n", 0x0B, status);
            }
            data = I2C_ReadOneByte(channelNumber * 2);
            data = data << 8;
            data = data | I2C_ReadOneByte(channelNumber * 2 + 1);
            data = data >> 2;
            temp = (float)data/32.0f;
            sum += temp;
        }
        temp = sum/4.0f;
        sum = 0.0f;

        printf("---读取芯片通道 %d  的温度---为: %.5f度 \n", channelNumber,
 temp);

        channelNumber += 1;
        if(channelNumber >4 )
        {
            channelNumber=0;
            printf("\r\n");
        }
    }

    //启动芯片写保护

    lock();
}
```

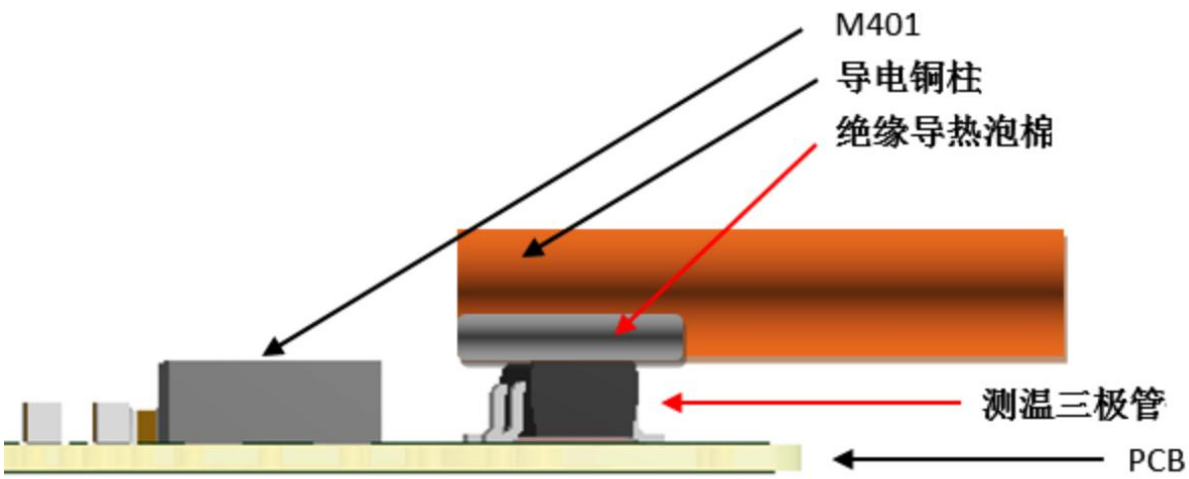
6.5 远端开短路检测（以#1 通道为例）

```
////////////////////////////////////////////////////////////////////////
void Check_Circuit(void)
{
    u8 status;
    s16 data = 0;
    float temp = 0.0f;


    //解除芯片写保护

    unlock();
```

```
    //远端#1 测温

    //设置测温通道为#1
    I2C_WriteOneByte(0x17, 0x15);
    I2C_WriteOneByte(0x18, 0x41);
    delay_ms(25);
    //读取状态寄存器
    status = I2C_ReadOneByte(0x0B);
    //读取状态寄存器的 BUSY 位以确定温度转换是否完成
    if(status&0x80)
    {
       printf("温度转换未完成！ REG 0x%02x --- Value ---0x%02x \r\n", 0x0
B, status);
    }
    data = I2C_ReadOneByte(2);
    data = data << 8;
    data = data | I2C_ReadOneByte(3);
    if(data&0x1)
    {
      printf("注意：测温电路开路。\n");
    }
    data = data >> 2;
    temp = (float)data/32.0f;
    if(temp == -256.0f)
    {
      printf("注意：测温电路短路。\n");
    }
    //启动芯片写保护
    lock();
}
```

七、 电表导热设计建议

导热泡棉本身的导热特性会影响测温，需要专门测试其热传导特性。具体可参考敏源传感导热泡棉测试相关文档——《M401 导热泡棉测试》。



导热泡棉测试结构示意图