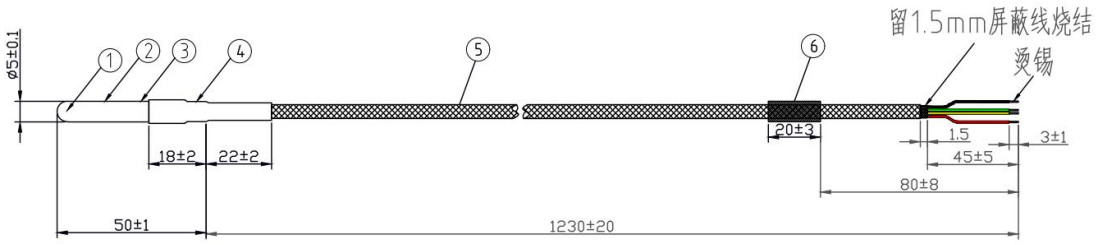
超低温高精度数字温度探头规格书

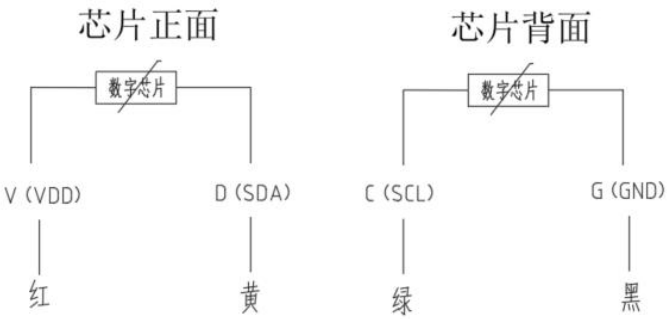
MTS4P100T5-FI2C-1200

MTS4P100T5-FI2C-1200 为超低温高精度数字温度探头，内部芯片型号为 MTS4P100，最高测温精度±0.3℃@-20℃~+40℃，工作温度范围-100℃~+150℃，电压范围 2.0V~5.5V，支持 I2C 通信。探头配置四芯镀银铁氟龙线，线长 1.2 米。

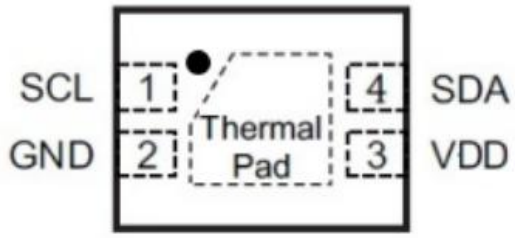
1. 探头规格示意图

单位：mm









线序	PIN 脚	说明	线色
1	SCL	I2C 时钟线	绿色
2	GND	地	黑色
3	VDD	电源	红色
4	SDA	I2C 数据线	黄色

序号	名目	规格
1	数字温度芯片	MTS4P100
2	不锈钢直管	304 不锈钢，Φ5.0*50mm
3	耐低温封装树脂	柔软性灌封料封装
4	热缩管	白色，Φ5.0*40mm（无字）
5	镀银四芯屏蔽线	整线外径Φ3mm，AWG=26#/1230±20mm
6	热缩管	透明，Φ3.0*20mm（无字）

2. 特性

- 芯片型号：MTS4P100
- 测温精度：±0.3℃@-20℃~+40℃
  ±0.5℃@-50℃~+85℃
  ±1℃@-100℃~+100℃
- 工作温度范围：-100℃~+150℃
- 工作电压范围：2.0V~5.5V
- 通信接口：I2C
- 防水等级：IP67
- 定制线长 1.2 米

注：其他指标及应用参见 MTS4 产品手册。

3. 应用指南

该产品全温进行了高阶拟合，16bit 系数 A、B、C、D 的存储位置分别为 User_define_0、User_define_1；User_define_2、User_define_3；User_define_4、User_define_5；User_define_6、User_define_7 一共 8 个字节，系数所在数组一共 15 个字节，第 15 个字节是前 14 个字节的 CRC 校验码，高阶拟合计算公式如下（t 为读温原始值，T 为高阶拟合温度值）：

$$T = A * (t)^3 + B * (t)^2 + C * t + D$$

计算温度时，需要在芯片驱动程序中加入下列代码（本示例底层函数与官网 MTS4 通用驱动例程(下载链接：https://gitee.com/mysentech/driver/tree/i2c_driver/T117_MTS4)中的例程通用）。

```
/**-------------------------------------------------------------------------
   * @brief  读取校准系数
   * @param  addr：IIC 地址
   * @param  Anum：系数 A
   * @param  Bnum：系数 B
   * @param  Cnum：系数 C
   * @param  Dnum：系数 D
   * @retval 无
-------------------------------------------------------------------------*/
typedef enum
{
    IIC_TI_lsb                  = 0x09,
```

```
    IIC_Tl_msb                  = 0x0A,
    EE_Cmd                      = 0x17,
    IIC_User_define_0           = 0x0C,
    IIC_User_define_1           = 0x0D,
    IIC_User_define_2           = 0x0E,
    IIC_User_define_3           = 0x0F,
    IIC_User_define_4           = 0x10,
    IIC_User_define_5           = 0x11,
    IIC_User_define_6           = 0x12,
    IIC_User_define_7           = 0x13,
    IIC_User_define_8           = 0x14,
    IIC_User_define_9           = 0x15,
    IIC_reserved1               = 0x69,
    IIC_reserved2               = 0x6A,
    IIC_reserved3               = 0x6B,
    EE_RESET                    = 0x6A,    //软复位,装载 EE 值到寄存器,与 EE 对应的部分，寄存器值恢复到 EE 保存值，不与 EE 对应的部分，寄存器值恢复到默认值

} REG;
void MTS4_IIC_Read_ABCD(uint8_t addr, double * Anum, double * Bnum, double * Cnum, double * Dnum)
{
    uint8_t ABCD[15]={0},Alsb,Amsb,Blsb,Bmsb,Clsb,Cmsb,Dlsb,Dmsb;
    uint16_t Anum16,Bnum16,Cnum16,Dnum16;
    int16_t Anumint16,Bnumint16,Cnumint16,Dnumint16;
    T117I2C_Receive(addr, IIC_User_define_0,&ABCD[0]);
    T117I2C_Receive(addr, IIC_User_define_1,&ABCD[1]);
    T117I2C_Receive(addr, IIC_User_define_2,&ABCD[2]);
    T117I2C_Receive(addr, IIC_User_define_3,&ABCD[3]);
    T117I2C_Receive(addr, IIC_User_define_4,&ABCD[4]);
    T117I2C_Receive(addr, IIC_User_define_5,&ABCD[5]);
    T117I2C_Receive(addr, IIC_User_define_6,&ABCD[6]);
    T117I2C_Receive(addr, IIC_User_define_7,&ABCD[7]);
    T117I2C_Receive(addr, IIC_User_define_8,&ABCD[8]);
    T117I2C_Receive(addr, IIC_User_define_9,&ABCD[9]);
    T117I2C_Receive(addr, IIC_Tl_lsb, &ABCD[10]);
    T117I2C_Receive(addr, IIC_Tl_msb, &ABCD[11]);
    T117I2C_Receive(addr, IIC_reserved1, &ABCD[12]);
    T117I2C_Receive(addr, IIC_reserved2, &ABCD[13]);
    T117I2C_Receive(addr, IIC_reserved3, &ABCD[14]);
    if(ABCD[14] != MY_CRC8(ABCD, 14))
    {
        PR("CRC ERROR,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X\r\n", ABCD[0],ABCD[1],ABCD[2],ABCD[3],ABCD[4],ABCD[5],ABCD[6],ABCD[7],ABCD[8],ABCD[9], ABCD[10],ABCD[11],ABCD[12],ABCD[13],ABCD[14]);
    }
    else
    {
        Amsb=ABCD[0];
        Alsb=ABCD[1];
        Bmsb=ABCD[2];
        Blsb=ABCD[3];
        Cmsb=ABCD[4];
        Clsb=ABCD[5];
        Dmsb=ABCD[6];
        Dlsb=ABCD[7];

        Anum16=(unsigned short)Amsb<<8|Alsb;
        Anumint16=(short)Anum16;
        *Anum=(double)Anumint16/10000000000;
```

```
            Bnum16=(unsigned short)Bmsb<<8|Blsb;
            Bnumint16=(short)Bnum16;
            *Bnum=(double)Bnumint16/10000000;

            Cnum16=(unsigned short)Cmsb<<8|Clsb;
            Cnumint16=(short)Cnum16;
            *Cnum=(double)Cnumint16/100000+1;

            Dnum16=(unsigned short)Dmsb<<8|Dlsb;
            Dnumint16=(short)Dnum16;
            *Dnum=(double)Dnumint16/1000;
        }
}
/**------------------------------------------------------------------------
   * @brief  测量读取温度
   * @param  addr：IIC 地址
   * @retval 无
------------------------------------------------------------------------*/
#include "math.h"
void MY_Convert(uint8_t addr)
{
        float Temp_f,Temp,fTemp; uint16_t Temp_u16;
        double Anum,Bnum,Cnum,Dnum;
        /*测温*/
        MY_ConvertTemp();
        MY_DELAY_MS(tCon_A32);//与配置的 Temp_Cfg 中的 avg 参数相关，Temp_Cfg 默认值为
0x79，AVG=32
        /*读温*/
        MY_ReadTempWaiting(&Temp_u16);
        Temp = MY_OutputtoTemp((int16_t)Temp_u16);
        MTS4_IIC_Read_ABCD(addr, & Anum, & Bnum, & Cnum, & Dnum);
        Temp_f =Anum*pow(Temp,3)+Bnum*pow(Temp,2)+Cnum*Temp+Dnum;
        printf("\nTemp: %.2f ", Temp_f);
}
```