磁电式振动位移变送器SDJ-701L

产品名称	磁电式振动位移变送器SDJ-701L
公司名称	上海旋机自动化技术有限公司
价格	.00/件
规格参数	
公司地址	上海市青浦区崧泽大道6638弄15号15幢529室
联系电话	021-51078867 18930732303

产品详情

磁电式振动位移变送器SDJ-701L将振动速度传感器、精密测量电路集成在一起，构成高精度振动测量系统，实现了传统的“ 传感器+监测仪表模式的振动测量系统的功能，该变送器可直接连接DCS、PLC或其它设备，是风机、电动机、水泵等工厂设备振动测量的理想选择。 技术参数供电电源：24VDC±10% 输入信号：取自内置振动速度传感器的信号灵 敏 度：20mv/mm/s±5%频率响应：10 ～1000 Hz或者5～1000 Hz（特殊说明）量　程：0-20mm/s(真有效值)　　0-200um(峰-峰值）测量误差：±1%满量程输出电流：4～20mA输出阻抗：≤500Ω温度范围：运行时：-25℃～+65℃储存时：-40℃～85℃相对湿度：至95%，不冷凝外形尺寸：φ33×75mm重量：约340g订货代号XJ-9200A□□（可选）-□□(V/D)- A□□- B□□-C□□选型说明可选：防水接头□：F-防水接头凯装出线□：B-凯装管必选：选型说明量程范围□□：振动速度量　10V-0～10mm/s；20V*-0～20mm/s； 30V-0～30mm/s；……振动位移量　100D-0～100μm；100D-0～200μm； 300D-0～300μm；……安装方向A□：1-水平；2-垂直；3*-通用安装螺纹B□：1*-M10×1.5；2-M8×1.25；3-磁座；4-特殊定做电缆长度C□：1-1m；2*-2m；3-3m；……无特殊情况，厂家按项生产；如有特殊要求，请与我公司协商选型举例：XJ-9200A-20V-A3-B1-C2磁电式振动位移变送器SDJ-701L

MAX31856热电偶温度变送器驱动设计与实现

在我们的产品中经常有需要温度检测的地方，而热电偶温度检测电路是我们常用的。热电偶温度检测的方法很多，有时出于简单方便的考虑我们会选择热偶温度变送器来实现，这一篇我们就来讨论使用MAX31856热电偶温度变送器实现温度的检测。

1、功能概述

MAX31856可以对任何类型热电偶的信号进行冷端补偿和数字转换，输出数据以摄氏度为单位。转换器温度分辨率达0.0078125 ° C，允许读取+1800 ° C、-210 ° C(取决于热电偶类型)的温度读数，热电偶电压测量精度达±0.15%。热电偶输入端提供±45V过压保护。

MAX31856内部的查找表(LUT)储存不同类型热电偶(K、J、N、R、S、T、E和B)的线性修正数据。而且MAX31856还具备50H和60Hz电网频率滤波，也是热电偶故障检测频率。SPI兼容接口允许选择热电偶类型并设置转换和故障检测过程。热电偶功能是检测热电偶线两端的温度差。可在热电偶的额定工作温度范围内测量其检测端(常称为“热”端)，关于热电偶测温范围：

MAX31856将冷端温度数据储存在寄存器0Ah和0Bh。使能冷端温度检测时，这些寄存器为只读，其中包含实测冷端温度加冷端失调寄存器的数值。冷端温度检测使能时，读取寄存器操作将DRDY引脚复位为高电平。应通过多字节传输读取该寄存器的两个字节，以确保两个字节的数据来自同一次温度更新。禁止冷端温度检测时，这些寄存器为可读/写寄存器，其中包含的实测温度值。如果需要，禁止内部冷端检测时，可将来自外部温度传感器的数据写入这些传感器。冷端温度钳位在128°C，小温度钳位在-64°C。

由于所有热电偶都具有非线性，必须对冷端补偿后的原始值进行线性修正，并转换为温度值。为实现这一处理，利用LUT产生经过线性化和冷端补偿的温度值；每次转换后，将其作为19位数据储存在线性化热电偶温度寄存器(0Ch、0Dh和0Eh)中。应通过多字节传输读取全部三个字节，以确保所有数据来自于同一次数据更新。关于线性化热电偶温度数据格式，

与MAX31856的通信通过16个包含转换、状态和配置数据的8位寄存器实现，全部设置均通过选择相应寄存器单元的对应地址完成，寄存器存储器映射所示为温度、状态和配置寄存器的地址。存取寄存器时，使用地址0Xh为读操作，地址8Xh为写操作。读写数据时，寄存器MSB在前。如果对只读寄存器执行写操作，不改变该寄存器的值。

2、驱动设计与实现

我们了解了MAX31856热偶温度变送器的基本情况，接下来我们考虑如何实现MAX31856热偶温度变送器的驱动程序。

2.1、对象定义

我们依然是基于对象的概念来实现驱动程序的设计。
所以我们首先来考虑对象类型的定义。作为一个对象至少包含有属性和操作。
我们先来分析一下MAX31856热偶温度变送器对象的属性有哪些。 MAX31856热偶温度变送器拥有16个寄存器，这些寄存器标识了MAX31856热偶温度变送器当前时刻所处的状态，所以我们将它们定义为属性。同时考虑到记录当前时刻读取的温度转换值和根据物理量程转换的温度值，所以我们将目标温度及冷端温度的ADC转换值及物理量值作为MAX31856热偶温度变送器对象的属性。接下来我们考虑一下MAX31856热偶温度变送器对象需要实现哪些操作。 我们只考虑与具体平台依赖性较强的操作。 对于MAX31856热偶温度变送器对象，当其完成AD转换回给出一个就绪指示，我们需要实时的检测这个信号，并且这个过程依赖于具体的软硬件平台，所以我们将检测过程设计为对象的操作。 我们使用MAX31856热偶温度变送器需要对其进行读写，这一过程也同样依赖于具体的软硬件平台，所以我们也将其作为对象的操作。另外MAX31856用一个片选信号，在实现总线操作时我们需要以此来选择目标器件，所以我们也将其作为对象的操作。 根据前述对属性和操作的分析，我们可以定义对象类型如下：

```
/*定义MAX31856对象类型*/typedef struct Max31856Object {    uint8_t regValue[16];    ui
nt32_t mDataCode;    uint32_t rDataCode;    float mTemperature;    //TC测量温度
    float rTemperature;    //冷端温度    uint8_t (*Ready)(void);    void (*R
```

```
eadData)(uint8_t *rData,uint16_t rSize);  void (*WriteData)(uint8_t *wData,uint16_t wSize);
  void (*ChipSelcet)(Max31856CSType cs);  //片选信号}Max31856ObjectType;
```

我们已经定义了对象类型，使用对象类型l可以声明类型变量，但类型变量必须要初始化才能使用，所以我们还需要设计一个对象的初始化函数。 在这个初始化函数中，我们需要将对象变量以及具体应用相关的属性及操作作为参数传入，并对对象的各个属性及操作函数指针赋初值。 具体实现如下；

```
/*初始化MAX31855对象*/void Max31856Initialization(Max31856ObjectType *tc,       //MA
X31856对象变量                    Max31856Ready ready,
        //就绪信号                    M
ax31856ReadDataType read,    //读MAX31856函数指针
        Max31856WriteDataType write,  //写MAX31856函数指针
              Max31856ChipSelcetType cs     //片选操作
函数指针                          ){  uint8_t re
gValue=0;  uint8_t rData[16]={0};  if((tc==NULL)||(ready==NULL)||(read==NULL)||(write=
=NULL))  {     return;  }  tc->Ready=ready;  tc->ReadData=re
ad;  tc->WriteData=write;  if(cs!=NULL)  {     tc->ChipSelcet=cs;
 }  else  {     tc->ChipSelcet=DefaultChipSelect;  }  tc->mD
ataCode=0;  tc->rDataCode=0;  tc->mTemperature=0.0;  tc->rTemperature=0.0;
 tc->ChipSelcet(Max31856CS_Disable);  regValue=0x81;  WriteRegister(tc,REG_CR0,regValue
);  ReadRegister(tc,REG_CR0,rData,16);  for(int i=0;i<16;i++)  {     tc
->regValue[i]=rData[i];  }}2.2、对象操作
```

我们定义了对象类型并实现了初始化函数，接下来我们需要考虑要对MAX31856热偶温度变送器执行整么样的操作，毕竟得到数据才是我们的目的。 我们考虑到需要设置相应的寄存器以实现相应功能，同时也需要获取寄存器的值以得到设备状态，或者从MAX31856热偶温度变送器获取测量数据。我们先来看怎么读取寄存器的值。 我们读取寄存器的值用于判断MAX31856当前的运行状态，前面我们已经叙述过寄存器的地址及功能，而读寄存器的时序要求如下：

根据前面的描述及上述时序图的要求可以编写读寄存器的函数如下：

```
/*读寄存器操作*/static void ReadRegister(Max31856ObjectType *tc,uint8_t regAddr,uint8_t *rData,uint
8_t rSize){  uint8_t wData=regAddr;  if(rSize<1)  {     return;
}  tc->ChipSelcet(Max31856CS_Enable);  tc->WriteData(&wData,1);  tc->ReadData(r
Data,rSize);  tc->ChipSelcet(Max31856CS_Disable);}
```

同样我们写寄存器时，我们根据前述寄存器的相关描述和写寄存器的时序图来实现。
写寄存器的时序图如下：

根据上述描述我们可以实现写寄存器值的函数如下：

```
/*写寄存器操作*/static void WriteRegister(Max31856ObjectType *tc,uint8_t regAddr,uint8_t value){
  uint8_t wData[2];  if(regAddr>11)  {     return;  }  w
Data[0]=regAddr+0x80;  wData[1]=value;  tc->ChipSelcet(Max31856CS_Enable);  tc->
WriteData(wData,2);  tc->ChipSelcet(Max31856CS_Disable);}
```

我们使用MAX31856的目的就是为了得到温度测量数据，所以我们来看看如何读取温度数据。
温度转换值可以一次读取测量数据和冷端数据，其时序图如下：

根据上述描述我们一次性读取6个字节的数据，具体实现如下：

```
/*获取MAX31855测量数据*/void Max31856GetDatas(Max31856ObjectType *tc){  uint8_t rData[6
]={0};  if(tc->Ready())  {    if((tc->regValue[REG_CR0]&0x80) != 0x80)
   {      WriteRegister(tc,REG_CR0,0x81);      R
eadRegister(tc,REG_CR0,rData,1);      tc->regValue[REG_CR0]=rData[0];
 }    return;  }  ReadRegister(tc,REG_CJTH,rData,6);  tc->rData
Code=(rData[0]<<8)+rData[1];  tc->mDataCode=(rData[2]<<16)+(rData[3]<<8)+rData[4];  tc
->regValue[REG_SR]=rData[5];  tc->mTemperature=CalcMeasureTemperature(tc->mDataCode);
 tc->rTemperature=CalcColdEndTemperature(tc->rDataCode);}
```

3、驱动的使用

我们已经实现了MAX31856热偶温度变送器的驱动程序，这一节我们来使用该驱动程序实现一个简单应用，以验证驱动程序的正确性。

3.1、声明并初始化对象

首先我们需要使用前面定义的MAC31856热偶温度变送器对象类型声明一个对象变量。
在我们的系统中，总线上挂载了4个MAX31856，所以我们声明如下：

```
Max31856ObjectType tcObj[4];
```

声明的对象变量需要先初始化方可使用，而初始化函数有5个参数。
个参数是需要要初始化的对象变量的指针，而余下的4个参数则是平台相关的操作函数指针。
这些函数的原型定义如下：

```
typedef uint8_t (*Max31856Ready)(void);typedef void (*Max31856ReadDataType)(uint8_t *rData,uint16
_t rSize);typedef void (*Max31856WriteDataType)(uint8_t *wData,uint16_t wSize);typedef void (*Ma
x31856ChipSelcetType)(Max31856CSType cs);  //片选信号
```

这几个函数则是我们需要根据具体的软硬件平台来实现的。 由于是在同一总线上，所以读写函数只需统一定义就好，但偏选信号和就绪信号则需根据模块单独定义。 具体的函数实现如下：

```
/*温度模块1就绪操作函数*/static uint8_t Tc1Ready(void){  uint8_t result=1;  result=HAL
_GPIO_ReadPin(TC1_RDY_GPIO_Port,TC1_RDY_Pin);  return result;}/*温度模块2就绪操作函数*/
static uint8_t Tc2Ready(void){  uint8_t result=1;  result=HAL_GPIO_ReadPin(TC2_RDY_
GPIO_Port,TC2_RDY_Pin);  return result;}/*温度模块3就绪操作函数*/static uint8_t Tc3Ready(v
oid){  uint8_t result=1;  result=HAL_GPIO_ReadPin(TC3_RDY_GPIO_Port,TC3_RDY_Pin);
  return result;}/*温度模块4就绪操作函数*/static uint8_t Tc4Ready(void){  uint8_t resul
t=1;  result=HAL_GPIO_ReadPin(TC4_RDY_GPIO_Port,TC4_RDY_Pin);  return result;}/*SP
I1写数据操作*/static void WriteData(uint8_t *wData,uint16_t wSize){  HAL_SPI_Transmit(&hspi
1, wData, wSize, 1000);}/*温度模块1片选操作函数*/static void Tc1ChipSelcet(Max31856CSType cs){
  if(Max31856CS_Enable == cs)  {    HAL_GPIO_WritePin(TC1_CS_GPIO
_Port, TC1_CS_Pin, GPIO_PIN_RESET);    return;  }  HAL_GPIO_WriteP
in(TC1_CS_GPIO_Port, TC1_CS_Pin, GPIO_PIN_SET);}/*温度模块2片选操作函数*/static void Tc2Chi
pSelcet(Max31856CSType cs){  if(Max31856CS_Enable == cs)  {    HAL_
GPIO_WritePin(TC2_CS_GPIO_Port, TC2_CS_Pin, GPIO_PIN_RESET);    return;
}  HAL_GPIO_WritePin(TC2_CS_GPIO_Port, TC2_CS_Pin, GPIO_PIN_SET);}/*温度模块3片
选操作函数*/static void Tc3ChipSelcet(Max31856CSType cs){  if(Max31856CS_Enable == cs)
  {    HAL_GPIO_WritePin(TC3_CS_GPIO_Port, TC3_CS_Pin, GPIO_PIN_RESE
```

```
T);      return;   }   HAL_GPIO_WritePin(TC3_CS_GPIO_Port, TC3_CS_Pin,
 GPIO_PIN_SET);}/*温度模块4片选操作函数*/static void Tc4ChipSelcet(Max31856CSType cs){
 if(Max31856CS_Enable == cs)   {      HAL_GPIO_WritePin(TC4_CS_GPIO_Port,
 TC4_CS_Pin, GPIO_PIN_RESET);      return;   }   HAL_GPIO_WritePin(TC
4_CS_GPIO_Port, TC4_CS_Pin, GPIO_PIN_SET);}
```

完成这些函数的定义后我们就可以初始化对象变量了！
将对象变量的指针以及这些函数的指针作为参数传递给初始化函数，具体实现如下：

```
/*初始化MAX31856对象*/   Max31856Initialization(&tcObj[3],
        Tc4Ready,                  ReadDa
ta,                  WriteData,
          Tc4ChipSelcet
    );   Max31856Initialization(&tcObj[0],
    Tc1Ready,                  ReadData,
                WriteData,
      Tc1ChipSelcet
);   Max31856Initialization(&tcObj[1],
 Tc2Ready,                  ReadData,
            WriteData,
   Tc2ChipSelcet                    );
Max31856Initialization(&tcObj[2],                  Tc3Ready,
                  ReadData,
         WriteData,                  Tc
3ChipSelcet                    );
```

至此我们就完成了对象变量的声明及初始化，在后续操作中就可以使用对象变量对对应的MAX32856热偶温度变送器进行各种操作。

3.2、基于对象进行操作

现在我们就可以在应用中使用驱动程序完成我们想要对MAX31856进行的操作了！在这个例子中我们分别读取4个MAX31856对象去测量值，并对这个测量值进行滤波处理。

```
Max31856GetDatas(&tcObj[0]);   tFilter[0].newValue=tcObj[0].mTemperature;   if(tcObj[0].mTe
mperature<0.0)   {      kF[0]=1.125;   }   aPara.phyPara.tc1Temp=Power3P
olyfit(BandSmoothingFilter(&tFilter[0]),0.0,0.0,kF[0],0.0);   Max31856GetDatas(&tcObj[1]);   tFilt
er[1].newValue=tcObj[1].mTemperature;   if(tcObj[1].mTemperature<0.0)   {
kF[1]=1.126;   }   aPara.phyPara.tc2Temp=Power3Polyfit(BandSmoothingFilter(&tFilter[1]),0.0,0.0
,kF[1],0.0);   Max31856GetDatas(&tcObj[2]);   tFilter[2].newValue=tcObj[2].mTemperature;
 if(tcObj[2].mTemperature<0.0)   {      kF[2]=1.125;   }   aPara.phyPar
a.tc3Temp=Power3Polyfit(BandSmoothingFilter(&tFilter[2]),0.0,0.0,kF[2],0.0);   Max31856GetDatas(&tcO
bj[3]);   tFilter[3].newValue=tcObj[3].mTemperature;   if(tcObj[3].mTemperature<0.0)   {
     kF[3]=1.125;   }   aPara.phyPara.tc4Temp=Power3Polyfit(BandSmoothingFilte
r(&tFilter[3]),0.0,0.0,kF[3],0.0);
```

到这里我们就完成了整个测试程序的编写，运行后能够正确读取温度数据，说明我们设计的驱动程序是正确的。

4、应用总结

在这一篇中我们设计并实现了MAX31856热偶温度变送器的驱动程序，也编写了测试应用来验证这一驱动程序，测试的结果符合我们的预期。事实上，这一测试应用是从我们的实际项目中提取出来的，我们设计的MAX31856热偶温度变送器驱动程序在实际项目中运行也完全符合要求。在使用驱动程序时需要注意，在我们的应用中是一条SPI总线挂载了4个MAX31856模块，所以需要偏选信号。如果在应用中MAX31856是硬件设定的偏选信号，则可以在初始化时使用NULL或者空函数替代。