SV SensTech

华景传感科技

Data Sheet

Version 1.0/Oct 2022

SV-SOP6-040D

拥有核心芯片技术的MEMS传感技术公司

A MEMS Sensor Company with Advanced Core Chip Technology

上海
芯片研发：上海张江

无锡
研发测试中心：无锡高新区

北京
华北销售中心：北京海淀

德国
芯片研发：斯图加特

苏州
封测生产：苏州高新区

深圳
华南销售中心：深圳南山

产品规格书

SV- SOP6-040D 表压压力传感器

● 产品特点

- 压力类型：表压&负压
- 压力范围：-40~40kPa
- 待机电流：＜0.1μA
- 最低功耗：6uA@1Hz
- 数字I2C接口：气压ADC分辨率24 Bit，温度ADC分辨率16 Bit
- 可靠性高、稳定性好、长期漂移低

● 产品应用

- 水位压力测试
- 表压传感器系统
- 电子血压计
- 医疗仪器
- 工业/工程气压控制

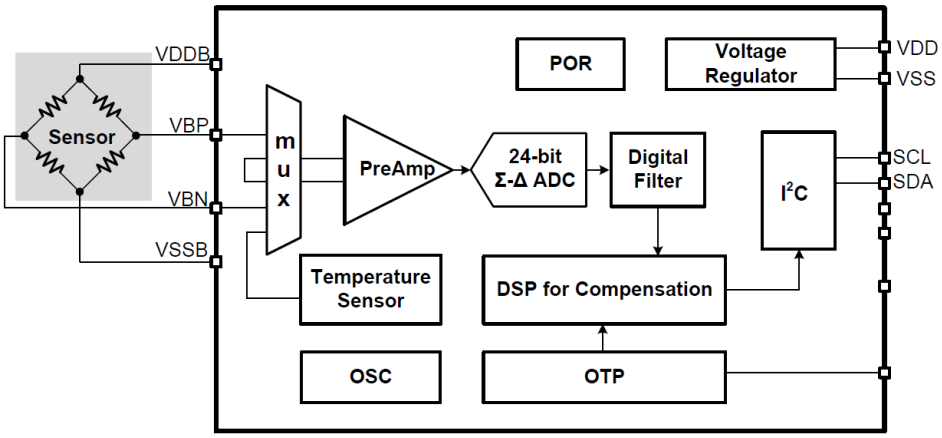
● 产品描述

SV-SOP6-040D 表压压力传感器是由本公司自主研发生产的扩散硅压阻式压力传感器芯片和专用信号处理集成电路芯片构成，每个模块都经过线性校验和温度补偿，使其输出信号线性正比于输入压力。

SV-SOP6-040D 表压压力传感器设计灵活，属于高精度和低电流消耗的小型化数字式气压传感器，可测量压力和温度值。每个压力传感器出厂前已被单独校准压力和温度并包含校准系数。在应用中使用系数将测量结果转换成真实的压力和温度值。被广泛应用于医疗仪器、电子血压计等领域。

目录

1.器件内部功能框图



2.极限参数

参数	条件	最小值	典型值	最大值	单位
存储温度		-40		105	℃
供电电压	所有的引脚	-0.3		3.6	V
IO 引脚电压	所有的引脚	-0.3		3.6	V
ESD	HBM,R=1.5kohm,C=100pF	-2		2	kV
过载				200	kPa

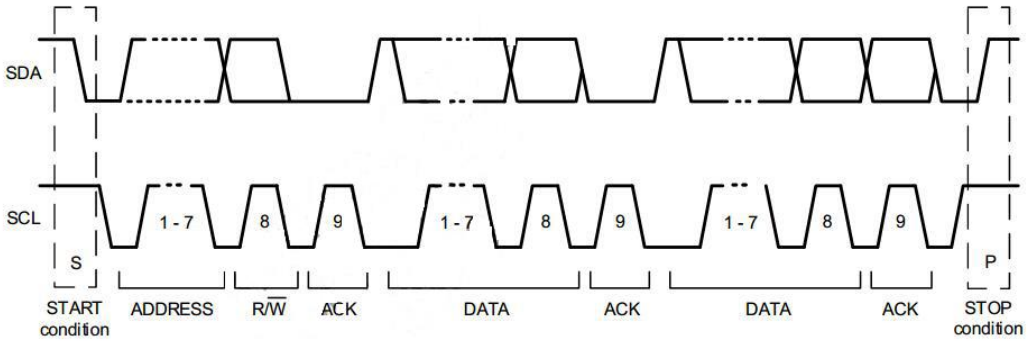
3.规格参数

（Vs=3.3VDC，T_A=25±3℃）

参数	符号	条件	最小值	典型值	最大值	单位
性能指标						
工作量程	Ps		-40		40	kPa
工作温度	T_A		-20		85	℃
	T_c	最佳工作范围	0	25	65	℃
压力输出精度	P_A	@-20~85℃	-1.0	±0.5	1.0	%F.S
压力噪声	$P_{\_noise}$			3.0		PaRMS
压力分辨率	$P_{\_res}$			2.0		Pa
温度漂移	-			3.0		Pa/℃

电源及电流消耗								
电源电压	VDD				1.8	3.3	3.6	V
电源抑制比	PSR_{VDD}	VDD=1.8V			17			dB
电流消耗（1 Hz）	I_{VDDAVP}	工作模式	OSR_P	OSR_T				
		低功耗模式	4X	4X		6	8	uA
		标准模式	16X			13	19	
		高精度模式	64X			42	50	
峰值电流	I_{PEAK}	峰值电流				0.3		mA
待机电流	I_{VDD}	25° C 时休眠状态的待机电流				50	100	nA
ADC 转换								
ADC 转换速率	FS,raw	OSR 为 2X~128X			20		1350	Hz
压力/温度测量时间	T_C	工作模式	OSR_P	OSR_T				
		低功耗模式	4X	4X		13		ms
		标准模式	16X			31		
		高精度模式	64X			105		
ADC 分辨率		压力				24		Bit
		温度				16		Bit
串行数据时钟	$Fc_{,I2C}$						3.4	MHz
温度传感器								
温度分辨率	T_{res}					0.003		K/LSB
温度范围	T_{txt_sp}	内置温度传感器			-40		150	℃
温度精度		0~65 ℃				±2		℃
上电时序								
启动时间	TSTA1	VDD 上升至接口开始通讯的时间					1	ms
	TSTA2	VDD 上升至开始测量的时间					2.5	ms
唤醒时间	TWUP1	休眠状态至接口开始通讯的时间					0.5	ms
	TWUP2	休眠状态至至开始测量的时间					2	ms

4. I2C 通讯协议



I2C 通讯协议

- START Condition

SDA 由空闲高状态转换为低状态，这是 SCL 保持高，这也能在传输过程中重复发送 start condition，这预示通讯将会重新开始而没有中间的停止位。

- Address Bits

在第一个字节传输过程中，前 7-bits 提供设备的指定地址，默认为 0x78，这个地址的设备将会应答本次的通讯。

- Read/Write Direction Bit

在第一个字节传输过程中，最后一比特指出通信方向，0 表示主设备写操作，1 表示主设备读操作，如果主设备请求读从设备，则主设备将在后来的字节控制 SDA 线输出数据。

- Data Byte

所有其他的字节，除了地址和读、写位，在 SDA 上传输被认为是通信的数据字节。

- Acknowledge or Not Acknowledge Bit

应答位用来告诉发送者字节已经接受到，设备收到数据需要应答每个字节，包括地址字节，在这个时刻，发送数据的总线设备停止驱动 SDA 线并且 SDA 线被拉高，不应答一个字节，接受设备不需要做任何事。应答一个字节，接受设备需要把 SDA 拉低。

一个接受从设备不需要应答，如果从设备不是寻址的设备或者设备不能处理接受的字节，主设备不应答，如果主设备在接受中并且想要结束通讯，如果遇到不应答，设备传输数据需要产生一个停止位。

- Stop Condition

SDA 从低状态转换到高状态，而且 SCL 保持高，这时结束 I²C 通信。

5.工作模式、操作顺序

产品上电后仅在接收到相应的 I²C 命令后才会启动一次压力和温度的测量及校准过程，完成测量后自动进入深度休眠状态以节省功耗。在深度休眠时，内部电路中除 I²C 接口外其余电路全部关闭，消耗的电流约为 0.1μA。

传感器 IIC 地址:传感器默认的 7bits I2C 设备地址为 0x78

操作顺序：

①上电

②IIC 写入

③等待

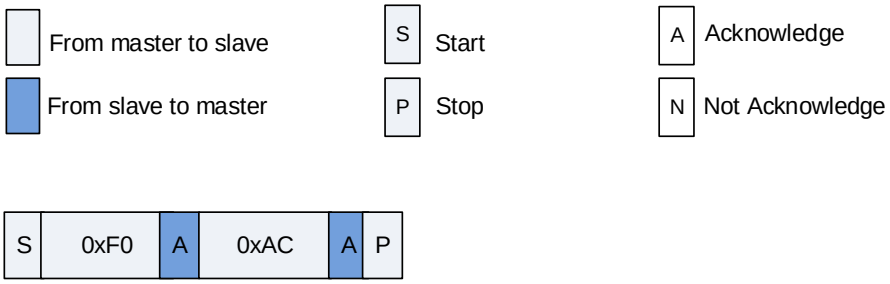
④IIC 读取

⑤数值换算

5.1 IIC 写入

发送 0xAC 命令可以获取到利用内部算法校准的数据。发送 0xAC 命令获取校准值的步骤如下：

5.1.1. 发送 Get_Cal 命令(0xAC)， 默认模式



图： 写命令

写命令中的 0xF0 表示默认的 7bits I2C 设备地址为 0x78，最后 1bit 为 0 表示主设备写操作。

Get_Cal 命令的过采样率是由 OTP 中的配置决定的，是固定不变的；

5.1.2 Get_Cal_S 命令

Get_Cal_S 命令(0Xb0 ~ 0xBF)与 Get_Cal 命令(0xAC)几乎一样，除了测量时 ADC 的过采样率的设定有所不同。Get_Cal_S 命令的过采样率是由命令本身的内容决定的。在用户校准完传感器，并烧写 OTP 数据之后，Get_Cal_S 命令允许用户用不同的过样率完成测量，以实现高精度、中等精度或低精度测量。

名称解释：过采样，类似于硬件滤波，数字越大，精度越好，需要消耗的时间越多，功耗也越高。

表：Get_Cal_S 命令

Command 0xBX(HEX)	Function	Detail
X 的第[3] Bit	测量温度时ADC的过采样率OSR_T	0: 4x 过采样率　　1: 8x 过采样率
X 的第[2:0] Bit	测量外部电桥时ADC的过采样率 OSR_P	000: 128X 过采样率　100: 8X 过采样率 001: 64X 过采样率　101: 4X 过采样率 010: 32x 过采样率　110: 2X 过采样率 011: 16X 过采样率　111: 1X 过采样率

5.2. 等待

发送完写命令后需要等待一段时间再发送读命令，因为内部完成整个测量需要一段时间。等待的时长取决于压力过采样率和温度过采样率的设置。

等待的时长不需要计算，可以通过不断的读 IIC 状态字的方式判断出是否采集已经完成。

不同的过采样率对应的等待时间不同,如下时间为 温度测量+压力测量的时间;

ADC 转换							
ADC 转换速率	FS,raw	OSR 为 2X~128X		20		1350	Hz
压力/温度 测量时间	Tc	OSR_P	OSR_T				ms
		2X	4X		10		
		4X			13		
		8X			19		
		16X			31		
		32X			56		
		64X			105		
		128X			203		

表：Status 字节比特位描述

比特位	意　义	描　述
Bit7	保留	固定为 0
Bit6	上电指示(Power indication)	1 设备上电（V_{DDB} on）；0 设备掉电
Bit5	忙闲指示 (Busy indication)	0 表明最近一次 I^2C 命令所要求读取的数据已经准备好被读取。 1 设备忙，表明最近一次 I^2C 命令所要求读取的数据还未有效。如果设备忙，新的命令将不被处理。
Bit4	保留	固定为 0
Bit[3]	工作状态 (Mode Status)	0 NOR mode(默认模式) 1 CMD mode
Bit2	存储器数据完整性指示 (Memory integrity/error flag)	0 表示 OTP 存储器数据完整性测试(CRC)通过。(默认模式) 1 表示完整性测试失败。对数据完整性的测试只在上电过程中(POR)计算一次，所以被写入的新 CRC 值只能在接下来的 POR 之后使用。
Bit1	保留	固定为 0
Bit0	保留	固定为 0

5.3. 读数

要保证写指令和读指令的时间间隔大于测量的时长才能够读出校准数据，读数格式如图 4 所示，读命令中的 0xF1 表示默认的 7bits I2C 设备地址为 0x78， 最后 1bit 为 1 表示主设备读操作。读到的校准数据共 6 个字节，依次为 1 字节状态字，3 字节气压相关值 BridgeDat[23:16],BridgeDat[15:8]，BridgeDat[7:0]

2 字节温度相关值:TempDat[15:8],TempDat[7:0]。

读值时候压力和温度要一起读取出来。如果不需要温度值，可以不进行温度的转换计算。

S	0XF1	A	Status	A	BridgeDat [23:16]	A	BridgeDat [15:8]	A	BridgeDat [7:0]	A	TempDat [15:8]	A	TempDat [7:0]	N	P

图：I2C 读出 3 字节压力相关值或 2 字节温度相关值

5.4. 换算

读到校准数据后，需要将以百分比形式表示的无符号数进行简单的换算。

为方便理解我们假设读到的校准数据为：0x04 0x9B 0xB0 0xC5 0x56 0xAA

0x04 为状态字 Bit5 为 1 表明最近一次 I2C 忙，需要等待一段时间。如果 Bit5 为 0 表明设备非忙，可以读取数据。关于状态字各比特的详细描述请参见附录。

0x9B 0xB0 0xC5 三个字节为电桥校准值

0x56 0xAA 两个字节为温度校准值

电桥校准值换算：将 0x9B 0xB0 0xC5 转换为十进制数为 10203333,

本次计算假设校准时使用的量程为-40~40kPa，对应的 AD 输出为 2516582.4~14260633.6（15%AD~85%AD）

根据 P2 输入输出关系校准公式得到：

实际压力值=（40+40）/（14260633.6-2516582.4）*（10203333-2516582.4）-40 = 12.362kPa

温度换算：将 0x56 0xAA 转换为十进制数为 22186，由于读取到的校准数据是以百分比形式表示的，这个百分比在数值上等于我们换算得到的十进制数与 16 bits 无符号数的最大值（65535）之比，所以在换算百分比时可进行如下计算

22186/65536*100% = 33.85%

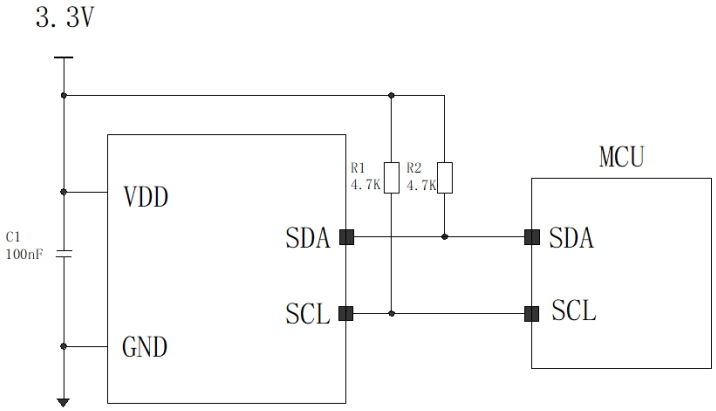
温度的校准范围规定为-40℃~150℃ 所以校准值=(150-(-40))*33.85% - 40 = 24.32℃

气压值换算：输入输出关系：

$$\boldsymbol{Pressure} = \frac{\mathbf{P_{MAX}} - \mathbf{P_{MIN}}}{\mathbf{D_{MAX}} - \mathbf{D_{MIN}}} \cdot (\mathbf{D_{test}} - \mathbf{D_{MIN}}) + \mathbf{P_{MIN}}$$

Pressure：实际压力值； Dtest：传感器的数字输出值； PMIN：传感器零点压力值； PMAX：传感器满量程压力值；DMIN：传感器零点时对应的数字输出值；DMAX：传感器满量程时对应的数字输出值；

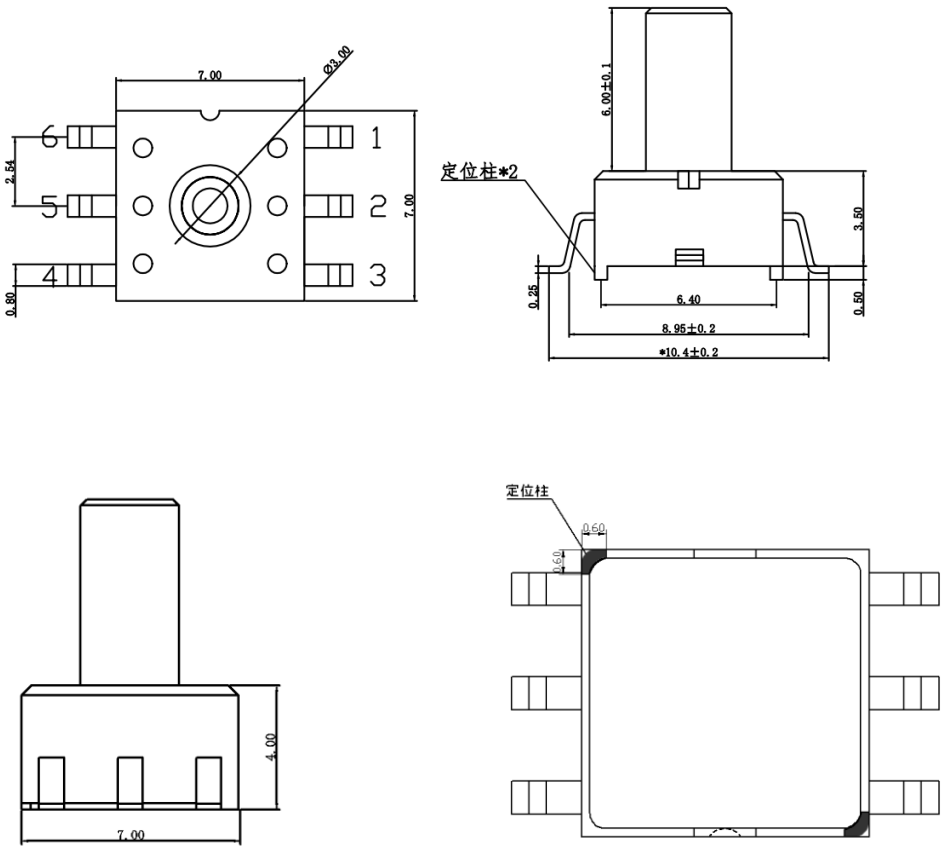
6.典型应用电路



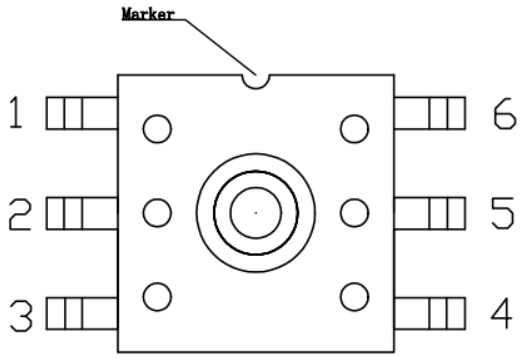
7. 机械特性

7.1 外形尺寸

单位：mm



7.2 引脚定义



引脚配置图

引脚	定义	说明
1	VDD	电源正极
2	SCL	I²C 通讯时序
3	SDA	I²C 通讯数据
4	NC	空置不接
5	GND	电源负极
6	NC	空置不接

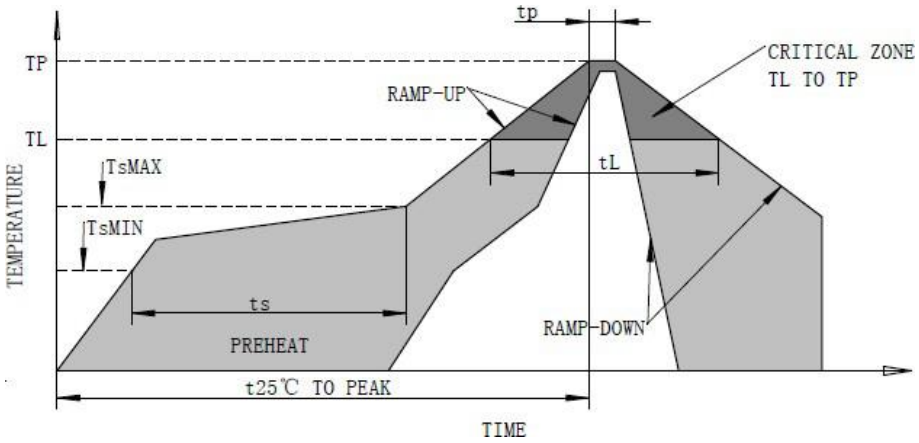
注： NC脚需保持悬空，否则可能会引起产品功能失效。

8. 注意事项

8.1. 焊接

8.1.1. 回流焊接(SMD 类型)

推荐回流焊接参数：



a） 平均升温率：MAX 3℃/S

b） 预热：最小温度（TsMIN）150℃， 最大温度（TsMAX）200℃，Ts 60~80S

c） 时间保持：温度（TL）217℃，时间（tL）60~150S

d） 峰值温度(TP)： 260℃， 25℃至峰值温度时间：MAX 8Min

e） 在实际峰值温度（TP）的时间（tP）：20~40S

f） 降温率：MAX 4℃/S

8.1.2. 波峰焊接（DIP 类型）

a） 预热30~120秒，温度控制在90~130℃，温度过低会导致焊接不良；

b） 焊接温度设置在240~260℃，焊接时间不超过5秒，时间过长，会出现尖端弯曲、连锡等现象；

8.1.3. 手工焊接

a）焊接时选用 25W 的外热式或 20W 的内热式电烙铁，设定温度 260~320℃，单点焊接不超过 5 秒。被焊金属面应保持清洁。氧化物和粉尘、油污等会妨碍焊料浸润被焊金属表面，在焊接前可用机械或化学方法清除；

b）焊接完成时，注意焊头移开的时机、方向和速度，避免产品移位以造成焊接不良的风险；

8.1.4. 焊接返修

a）返修压力传感器产品及其周围器件严禁使用热风枪，SMD 类型产品应选择使用点锡膏过回流的方式；

b）采用烙铁手工维修时，避免长时间接触传感器 PAD，以免造成产品故障或焊盘（包括 PCB 焊盘）脱落；

c）在接插件焊接过程中注意压力传感器的防护，不要直接用手触摸器件，以及防止焊锡落入器件内部。

8.2. 清洗

焊接完后尽可能用专用的印制板清洗剂清洗;

严禁对压力传感器进行清洗或液体擦拭，即使对其周围器件清洗或液体擦拭过程中也要注意不能有液体进入气压计内部，以造成输出异常;

在接插件焊接过程中注意压力传感器的防护，不要直接用手触摸器件，以及防止焊锡落入器件内部，避免产品发生故障;

8.3. 设计注意事项

a）压力传感器设计在远离应力集中区域;

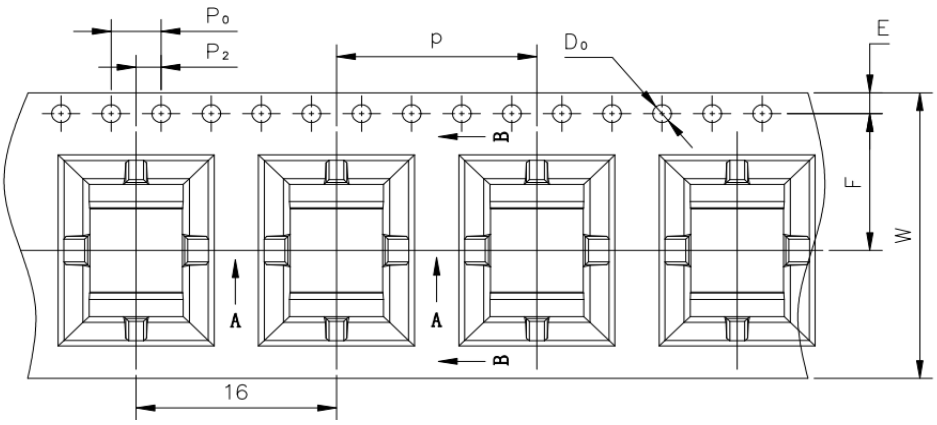
b）压力传感器设计远离热源器件（包括 PCB 双面）;

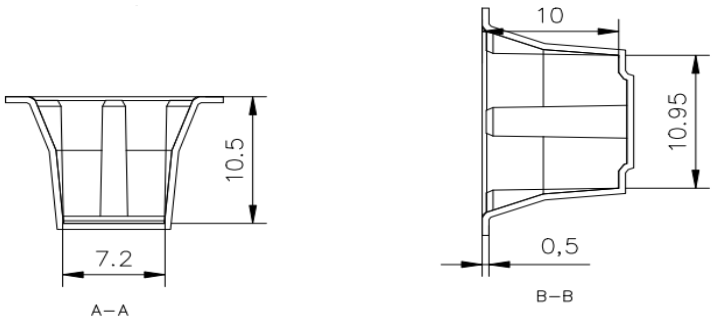
c）压力传感器不能被非透气材料包附，不可以将进气孔或透气孔堵住;

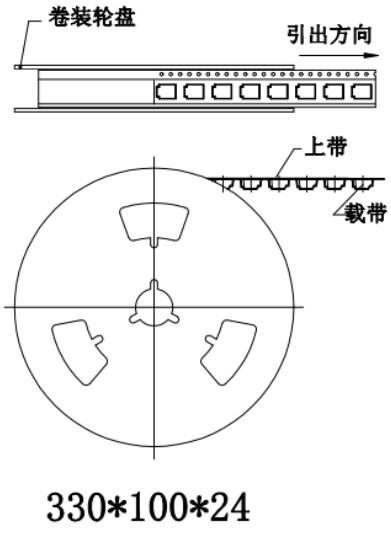
9. 包装规格

载带信息：

每卷数量：400 pcs 单位：mm







330*100*24

尺寸栏	
E	1.75±0.10
F	11.50±0.15
P2	2.00±0.15
ØDo	$1.50^{+0.10}_{-0.00}$
ØD1	
Po	4.00±0.10
10Po	40.00±0.20
W	24.00±0.30
P	16.00±0.10
Ao	7.20±0.15
B0	10.95±0.15
K0	10.50±0.15
t	0.50±0.05

10. 更改版本

版本号	变更内容	变更日期
1.0	新建	2022-10-28

11. 联系方式

华景传感科技（无锡）有限公司

地址：江苏省无锡市新吴区菱湖大道200号F2栋，214135

电话：(86) 0510-8562 2282

传真：(86) 0510-8562 2278

邮箱：sales@svsens.com

网站：www.svsens.com

12.附件：IIC 参考例程

```
#include "Sensor040D.h"
#include "Sensor040D _IIC.h"

//IIC clock line
sbit SCL = P1 ^ 1;

//IIC data line
sbit SDA = P1 ^ 0;

//Set the input and output mode of IIC data pin
#define Set_SDA_INPUT()
    P1MDOUT &= 0xFE;
    P1 |= 0X01
#define Set_SDA_OUTPUT() P1MDOUT |= 0x01;

////Delay function needs to be defined
void DelayUs(unsigned char i)
{
}

//Start signal
void Start(void)
{
    SDA = 1;
    DelayUs(2);
    SCL = 1;
    DelayUs(2);
    SDA = 0;
    DelayUs(2);
    SCL = 0;
}

//Stop signal
void Stop(void)
{
    Set_SDA_OUTPUT();
    SDA = 0;
    DelayUs(2);
    SCL = 1;
    DelayUs(2);
    SDA = 1;
    DelayUs(2);
}

//Read ACK signal
unsigned char Check_ACK(void)
{
    unsigned char ack;
    Set_SDA_INPUT();
    SCL = 1;
    DelayUs(2);
    ack = SDA;
```

```
    SCL = 0;
    Set_SDA_OUTPUT();
    return ack;
}

//Send ACK signal
void Send_ACK(void)
{
    Set_SDA_OUTPUT();
    SDA = 0;
    DelayUs(2);
    SCL = 1;
    DelayUs(2);
    SCL = 0;
    DelayUs(2);
}

//Send one byte
void SendByte(unsigned char byte)
{
    unsigned char i = 0;
    Set_SDA_OUTPUT();
    do
    {
        if (byte & 0x80)
        {
            SDA = 1;
        }
        else
        {
            SDA = 0;
        }
        DelayUs(2);
        SCL = 1;
        DelayUs(2);
        byte <<= 1;
        i++;
        SCL = 0;
    } while (i < 8);
    SCL = 0;
}

//Receive one byte
unsigned char ReceiveByte(void)
{
    unsigned char i = 0, tmp = 0;
    Set_SDA_INPUT();
    do
    {
        tmp <<= 1;
        SCL = 1;
        DelayUs(2);
        if (SDA)
        {
```

```
                tmp |= 1;
            }
            SCL = 0;
            DelayUs(2);
            i++;
        } while (i < 8);
        return tmp;
}

//Write a byte of data through IIC
uint8 BSP_IIC_Write(uint8 address, uint8 *buf, uint8 count)
{
        unsigned char timeout, ack;
        address &= 0xFE;
        Start();
        DelayUs(2);
        SendByte(address);
        Set_SDA_INPUT();
        DelayUs(2);
        timeout = 0;
        do
        {
            ack = Check_ACK();
            timeout++;
            if (timeout == 10)
            {
                Stop();
                return 1;
            }
        } while (ack);
        while (count)
        {
            SendByte(*buf);
            Set_SDA_INPUT();
            DelayUs(2);
            timeout = 0;
            do
            {
                ack = Check_ACK();
                timeout++;
                if (timeout == 10)
                {
                    return 2;
                }
            } while (0);
            buf++;
            count--;
        }
        Stop();
        return 0;
}

//Read a byte of data through IIC
uint8 BSP_IIC_Read(uint8 address, uint8 *buf, uint8 count)
```

```
{
    unsigned char timeout, ack;
    address |= 0x01;
    Start();
    SendByte(address);
    Set_SDA_INPUT();
    DelayUs(2);
    timeout = 0;
    do
    {
        ack = Check_ACK();
        timeout++;
        if (timeout == 4)
        {
            Stop();
            return 1;
        }
    } while (ack);
    DelayUs(2);
    while (count)
    {
        *buf = ReceiveByte();
        if (count != 1)
            Send_ACK();
        buf++;
        count--;
    }
    Stop();
    return 0;
}

// Define the upper and lower limits of the calibration pressure
#define PMIN -40000 // Zero Point Pressure Value for example -40kPa
#define PMAX 40000 // Full range pressure Value, for example 40kPa
#define DMIN 2516582.4 //AD value corresponding to pressure zero, for example 15%AD
#define DMAX 14260633.6 //AD Value Corresponding to Full Pressure Range, for example 85%AD

//The 7-bit IIC address of the Sensor Sensor040D is 0x78
uint8 Device_Address = 0x78 << 1;

//Delay function needs to be defined
void DelayMs(uint8 count)
{
}

//Read the status of IIC and judge whether IIC is busy
uint8 Sensor040D_IsBusy(void)
{
    uint8 status;
    BSP_IIC_Read(Device_Address, &status, 1);
    status = (status >> 5) & 0x01;
    return status;
}
```

```
void Sensor040D_get_cal(void)
{
	uint8 buffer[6] = {0};
	uint32 Dtest = 0;
	uint16 temp_raw = 0;
	double pressure = 0.0, temp = 0.0;

	//Send 0xAC command and read the returned six-byte data
	buffer[0] = 0xAC;
	BSP_IIC_Write(Device_Address, buffer, 1);
	DelayMs(5);
	while (1)
	{
		if (040D_IsBusy())
		{
			DelayMs(1);
		}
		else
			break;
	}
	BSP_IIC_Read(Device_Address, buffer, 6);

	//The returned pressure and temperature values are converted into actual values according to the calibration
range
	Dtest = ((uint32)buffer[1] << 16) | ((uint16)buffer[2] << 8) | buffer[3];
	temp_raw = ((uint16)buffer[4] << 8) | (buffer[5] << 0);
	pressure = (PMAX-PMIN)/(DMAX-DMIN)*(Dtest-DMIN)+PMIN;
	temp = (double)temp_raw / 65536;
	temp = temp * 19000 - 4000;
}
```