

GZP6859D

型压力传感器

数字输出

产品规格书

版本号： V1.6

文件发行日期： 2023.05.22

目录

1.产品描述.....	4
1.1 产品特点.....	4
1.2.应用领域.....	4
2.功能描述.....	4
2.1 引脚定义说明.....	5
2.2 框图.....	5
2.3 精度.....	6
3.技术指标.....	7
3.1 最大额定参数.....	7
3.2 性能指标.....	7
3.3 电气特性.....	8
4.应用电路.....	8
5.I ² C 通讯协议.....	9
6.寄存器描述.....	11
7.工作模式说明.....	12
8.外形结构.....	14
9.选型指南.....	15
10.常用量程型号.....	15
11.使用注意事项.....	17
11.1 焊接.....	17
11.2 清洗要求.....	18
11.3 存储与运输.....	18
11.4 其他使用注意事项.....	18
12.包装信息.....	19
安全注意事项	20
IIC Example Code (附件: IIC 代码案例).....	21

文件修订历史

修订	描述	日期
V1.0	初始版本	2020.12.21
V1.1	增加封面、目录	2021.11.05
V1.2	增加引脚定制选型	2022.03.11
V1.3	调整产品归类	2022.03.16
V1.4	修改供电信息，取消供电选型	2022.06.08
V1.5	更改公司地址	2022.08.29
V1.6	更新模板	2023.05.22

公司保留在不另行通知的情况下对其所包含的规格进行更改的权利。

产品规格书版权及产品最终解释权归芯感智所有。

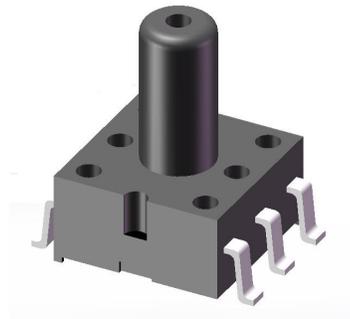
1.产品描述

GZP6859D 型压力传感器采用 SOP6 封装形式，内部集成了高精度 ADC 芯片，对传感器芯片输出的偏移、灵敏度、温漂和非线性进行数字补偿，以供电电压为参考，产生一个经过校准、温度补偿后的标准数字信号。

GZP6859D 型压力传感器尺寸小，方便客户安装。产品广泛用于医疗电子、汽车电子、运动健身器材等领域。

1.1 产品特点

- 测量范围-100kPa...0 ~ 5kPa...200kPa
- 表压型
- SOP6 封装
- 电源电压: 2.5V ~ 5.5V
- IIC 通讯



1.2.应用领域

- 电子血压计、呼吸机、制氧机、监护仪、雾化器等医疗领域
- 负压测量、压力仪表、气动元件等领域
- 按摩器、按摩椅、气垫床等运动健身器材领域
- 真空包装机、真空搅拌机、真空破壁机、真空保鲜盒、真空泵等真空负压领域
- 洗衣机、啤酒机、咖啡机、吸尘器、净水机、热水器等家电领域

2.功能描述

本产品用先进的微机电原理制作，核心技术为基于硅压阻效应的 MEMS 压力传感器芯片和高性能的信号调理 ASIC 芯片，硅微压阻式 MEMS 压力传感器 芯片通过四个应变敏感电阻构成的惠斯通电桥，输出信号被 ASIC 芯片放大、温度补偿和线性化，传递函数的线性化和温度补偿由 ASIC 中的数字处理电路实现，通过多项式补偿算法和多个温度下的多点压力标定技术，实现了全工作温度范围内的高精度压力测量。

2.1 引脚定义说明

压力传感器引脚配置如图 1。

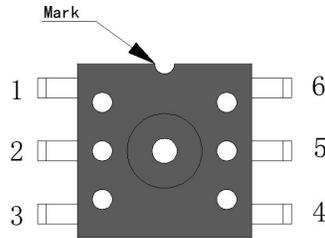


图 1. 引脚示意图

表 1. 引脚对应关系

序号	描述	备注
1	NC	悬空引脚
2	VDD	电源输入正极
3	NC	悬空引脚
4	SDA	I2C 模式下数据线
5	SCL	时钟信号
6	GND	电源输入负极

表 1

2.2 框图

传感器功能框图如图 2 所示

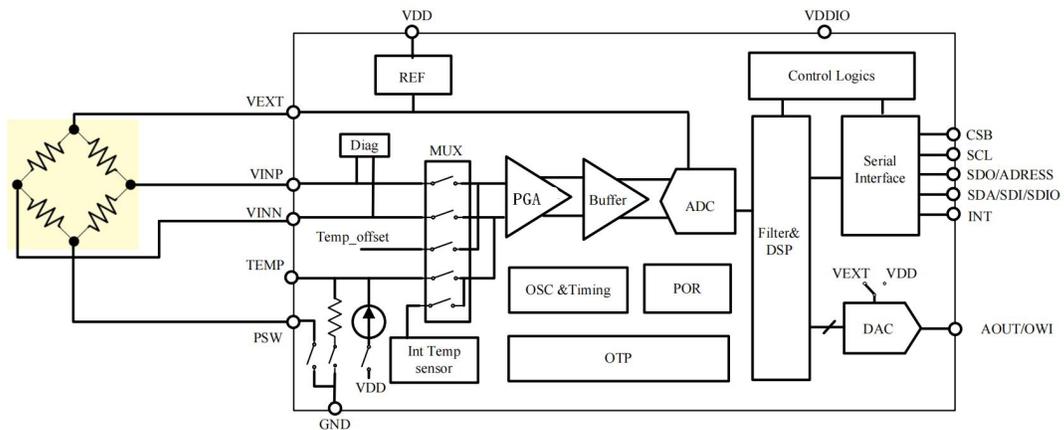


图 2 传感器功能框图

2.3 精度

GZP6859D 压力传感器的精度受由其线性、重复性、迟滞的误差组成。用传递函数计算的值是传感器的规定值，也是理论值。传感器的误差等于传感器在规定输入压力下的实际输出电压值与规定的输出电压值的差值。

2.4.1 综合误差

综合误差在产品的精度基础上，还包括更多的误差源：

压力漂移:指定压力范围内，零点和满量程的实际输出电压和规定输出电压的输出偏差。

温度效应:在温度范围内的，零点和满量程在不同温度下的输出偏差。

综合误差采用误差带表示，该误差带由三段线段组成，数据如图 4 和表 2 所示。

温度(°C)	综合误差(Fs)
-20~100	±3.0%
0~60	±1.5%

* 压力量程不同，综合误差不同，请咨询客服获取更多细节。

表 2 综合误差

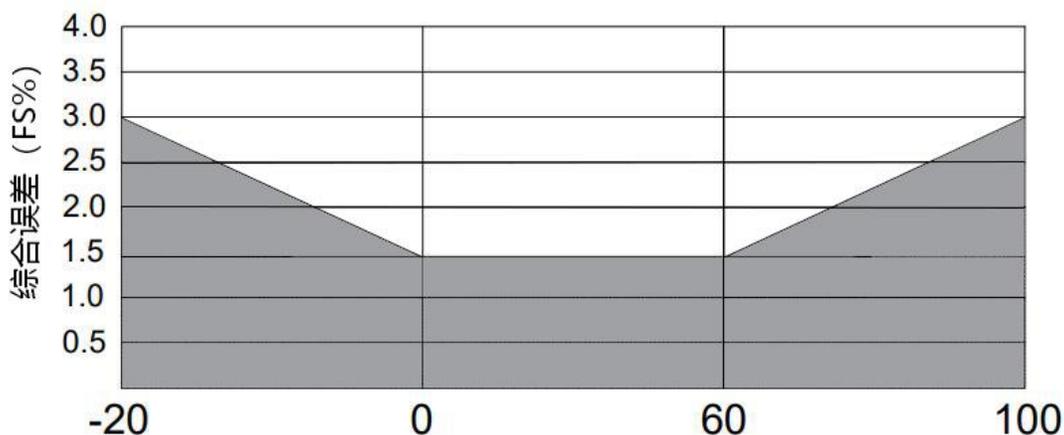


图 3 综合误差与温度关系

3.技术指标

传感器以下指标均在供电电源(5±0.25)V DC 和温度 25°C下测得

3.1 最大额定参数

传感器最大额定参数如表 3 所示

参数	最小值	典型值	最大值	单位	备注
供电电压	-0.3		6.5	V	
ESD 防护		4		kV	
储存温度	-30		150	°C	

表 3

3.2 性能指标

传感器性能指标如表 4 所示

项目	数值	单位
精度*	±1	%Span
响应时间	2.5ms@OSR_P=1024X	ms
SDA/SCL 上拉电阻	4.7	K ohm
ESD HBM	4000	V
零点温度漂移	±0.03	%FS/°C
满程温度漂移	±0.03	%FS/°C
过载压力	4× (量程≤60kPa) 2.5× (量程>60kPa)	Rated
破坏压力	5× (量程≤60kPa) 3× (量程>60kPa)	
补偿温度	0~60 (可定制)	°C
工作温度	-30~100	°C
贮存温度	-30~150	°C

* 精度为 0~60°C范围内的输出误差, 量程不同, 精度不同, 请咨询客服获取更多细节。

表 4

3.3 电气特性

传感器电气特性如表 5 所示

参数	最小值	典型值	最大值	单位	备注
供电电压	2.5	3.3	5.5	V	
待机电流		100		nA	
电流消耗		5		uA	一次测量
LDO 输出	1.62	1.8	1.98	V	3.3V 供电
PSRR		60		dB	
滤波电容		100		nF	
输出数据分辨率	23			Bits	$LSB=(1/2^{23}) * V_{EXT}$ ('raw_data_on' =1)
内置温度传感器		±1		°C	
输出数据解析度	16			Bit	$LSB = (1/256) °C$
时钟脉冲频率			400	KHz	I2C 通讯
IIC 设备地址		0x6D			

表 5

4.应用电路

芯片推荐应用电路见图 5

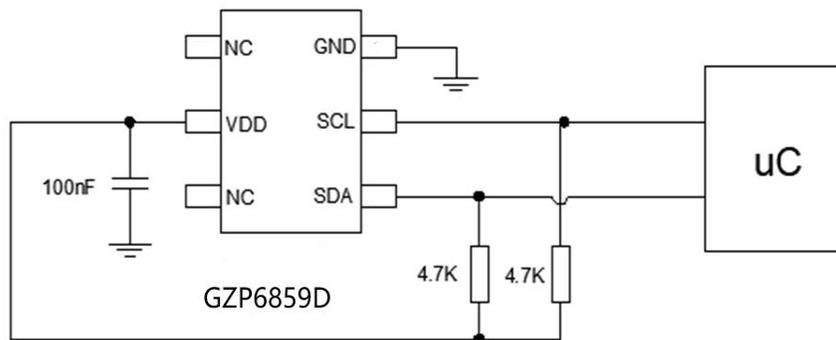


图 4

注意:

1. 装配前请确认好电气定义
2. NC 脚不要有任何的电气连接, 否则可能会造成产品功能失效
3. 焊装过程中做好防静电保护
4. 过载电压(6.5Vdc)可能烧毁电路芯片
5. 请在 VDD 和 GND 之间加上 0.1uf 电容
6. 本产品无反接保护, 装配时请注意电源极性

5. I²C 通讯协议

I2C 总线使用 SCL 和 SDA 作为信号线, 这两根线都通过上拉电阻 (典型值 4.7K) 连接到 VDD, 不通信时都保持为高电平。I2C 设备地址为 0x6D。

表 6. I2C 通讯引脚的电性特性

标示	参数	条件	最小值	最大值	单位
fsc1	时钟频率			400	KHz
tLOW	时钟低脉冲维持时间		1.3		us
tHIGH	时钟高脉冲维持时间		0.6		us
tSUDAT	SDA 建立时间		0.1		us
tHDDAT	SDA 保持时间		0.0		us
tSUSTA	每次开始时的建立时间		0.6		us
tHDSTA	开始条件保持时间		0.6		us
tSUSTO	停止条件建立时间		0.6		us
tBUF	两次通讯间隔时间		1.3		us

表 6

I2C 时序图

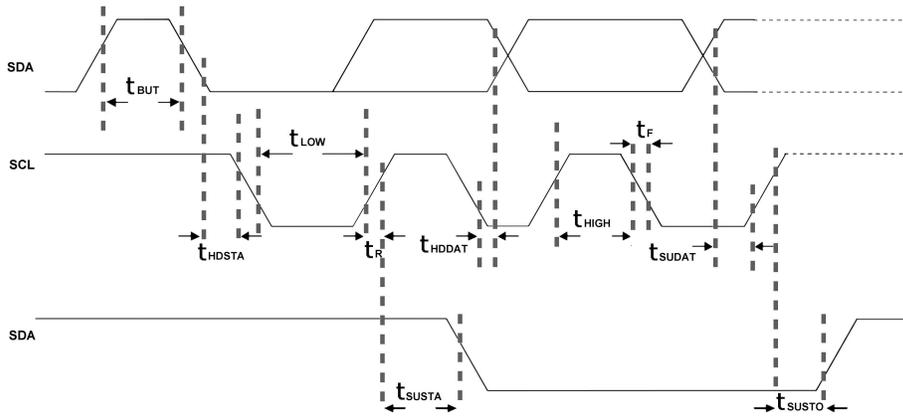


图 5. I2C 时序图

I2C 通讯协议有着特殊的开始(S)和终止(P)条件。当 SCL 处于高电平同时，SDA 的下降沿标志数据传输开始。I2C 主设备依次发送从设备的地址（7 位）和读/写控制位。当从设备识别到这个地址后，产生一个应答信号并在第九个周期将 SDA 拉低。得到从设备应答后，主设备继续发送 8 位寄存器地址，得到应答后继续发送或读取数据。SCL 处于高电平，SDA 发生一个上升沿动作标志 I2C 通信结束。除了开始和结束标志之外，当 SCL 为高时 SDA 传输的数据必须保持稳定。当 SCL 为低时 SDA 传输的值可以改变。I2C 通信中的所有数据传输以 8 位为基本单位，每 8 位数据传输之后需要一位应答信号以保持继续传输。

I2C 协议

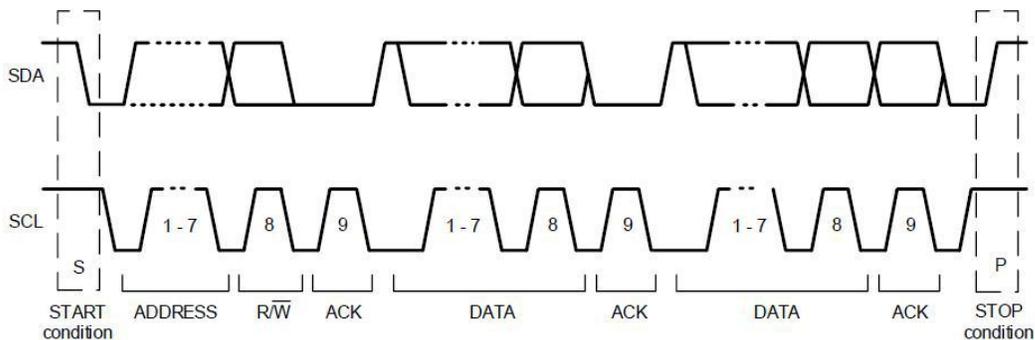


图 6.I2C 协议

6.寄存器描述

表 7.寄存器描述

地址	描述	R/W	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	默认值
0x06	DATA_MSB	R	Data out<23:16>								0x00
0x07	DATA_CSB	R	Data out<15:8>								0x00
0x08	DATA_LSB	R	Data out<7:0>								0x00
0x09	TEMP_MSB	R	Temp out<15:8>								0x00
0x0A	TEMP_LSB	R	Temp out<7:0>								0x00
0x30	CMD	RW	Sleep_time<7:4>			sco	Measurement_ctrl<2:0>			0x00	
0xA5	Sys_config	RW	Aout_config<7:4>			LDO_conf	Unipolar	Data_ou	Diag_on	OTP	
0xA6	P_config	RW		Input	Gain_P<5:3>			OSR_P<2:0>		OTP	

表 7

Reg0x06-Reg0x08 压力数据寄存器

Reg0x09-Reg0x0A 温度数据寄存器

Reg0x30 (测量命令寄存器) :

Measurement_control<2:0>(工作模式)

000,单次温度采集模式。

001,单次传感器压力信号采集模式。(使用此模式之前需要先读取温度,以获取温度校准系数,否则读数不准)

010,组合采集模式(一次温度采集后立即进行一次传感器压力信号采集)。

011,休眠模式(定期的执行一次组合采集模式,间隔时间由'sleep_time'决定)

Sleep_time<7:4>: 0001:62.5ms, 0010:125ms ... 1111: 1s, 0000:无意义。

(仅在休眠工作模式下有效)

Sco:数据采集完成标志位。1,开始数据采集;0,采集结束(休眠工作模式除外)。

Reg0xA5

Aout_config<7:4>:模拟输出配置 (建议保留默认配置)

LDO_config: 内部 LDO 配置。0, 配置成 1.8V; 1, 配置成 3.6V

Unipolar: 0, ADC 原始数据以有符号数格式输出; 1: ADC 原始数据以无符号格式输出。(仅当 'Data_out_control' =1 有效)

Data_out_control: 0, 输出校准数据; 1, 输出 ADC 原始数据 (默认配置为 0)

Diag_on: 0,关闭诊断功能; 1,开启诊断功能 (默认开启)

Reg0xA6

Input Swap:在传感器内部交换差分信号极性。

Gain_P<5:3>:采集传感器信号时 PGA 增益, 000:增益=1X。001:增益=2X。010:增益=4X。011:增益=8X。100: 增益=16X。101:增益=32X。110: 增益=64X。111:增益=128X。

OSR_P<2:0>:采集传感器信号时的过采样, 000:1024X, 001:2048X, 010:4096X, 011:8192X,100:256X, 101:512X, 110:16384X, 111:32768X。

7. 工作模式说明:

7.1. 组合数据采集模式

设置 'measurement_control' =010 和 'sco' =1 进入组合数据采集模式。

芯片上电后先后进行一次温度数据采集和一次传感器数据采集, 完成后回到待机模式, 并自动将

'sco' 置 0。在组合采集模式下, "Data_out_control" 寄存器必须设置为 0, 校准后的温度数据储存在 0x09~0x0A 寄存器, 压力数据储存在 0x06~0x08 寄存器。

7.2.休眠数据采集模式

设置 'measurement_control' =011 和 'sco' =1 进入休眠数据采集模式。芯片上电后，以一定的时间间隔进行一次温度数据采集和一次传感器数据采集，间隔时间由 'sleep_time' 设置，范围为 62.5ms 到 1s。除非手动将 'sco' 置 0，不然不会停止采集。在休眠数据采集模式下

'Data_out_control' 必须设置为 0，校准后的温度数据储存在 0x09~0x0A 寄存器，压力数据储存在 0x06~0x08 寄存器。

组合模式读取数据按照如下指令顺序进行操作：

- 1) 发送指令 0x0A 到 0x30 寄存器进行一次温度采集，一次压力数据采集。
- 2) 读取 0x30 寄存器地址，若 Sco 位为 0 代表采集结束，可以读取数据。或等待延迟 10ms。
- 3) 读取 0x06、0x07、0x08 三个寄存器地址数据构成 24 位 AD 值（压力数据 AD 值），读取 0x09、0x0A 两个寄存器地址数据构成 16 位 AD 值（温度数据 AD 值）
- 4) 按以下公式换算成实际压力、温度值：

· 最高位为 "0" 代表正压/正温度：

$$\text{Pressure} = \text{Pressure_ADC} / k; \quad \text{Temperature} = \text{Temp_ADC} / 256;$$

· 最高位为 "1" 代表负压/负温度：

$$\text{Pressure} = (\text{pressure_ADC} - 16777216) / k; \quad \text{Temperature} = (\text{Temp_ADC} - 65536) / 256;$$

注：1) 传感器校准后的输出可视为当前实际压力值 ($\pm 1\%$ Span)

2) 传感器校准后的输出：单位 Pa（默认），若要显示其他单位，可在换算公式里输入相应的系数进行换算；

3) 关于上述压力 ADC 换算公式中 k 值的选取可参照下表：

表 8.最大测量点 P 与系数 k 值对照表

最大测量点 P 值范围	k 值
$524 < P < 1000$	8
$262 < P \leq 524$	16
$131 < P \leq 262$	32
$65 < P \leq 131$	64
$32 < P \leq 65$	128
$16 < P \leq 32$	256
$8 < P \leq 16$	512
$4 \leq P \leq 8$	1024
$2 \leq P < 4$	2048

表 8

P 取测量点最大数值（绝对值），比如，测量-20~40kpa，P 取 40，因为 $32 < 40 \leq 65$ ，所以 k 值为 128；又比如，测量范围-100~50kpa，P 取 100，因为 $65 < P \leq 131$ ，所以 k 值为 64。

8.外形结构 (单位: mm)

传感器外型尺寸参照图 7 (未标注公差±0.1mm)。

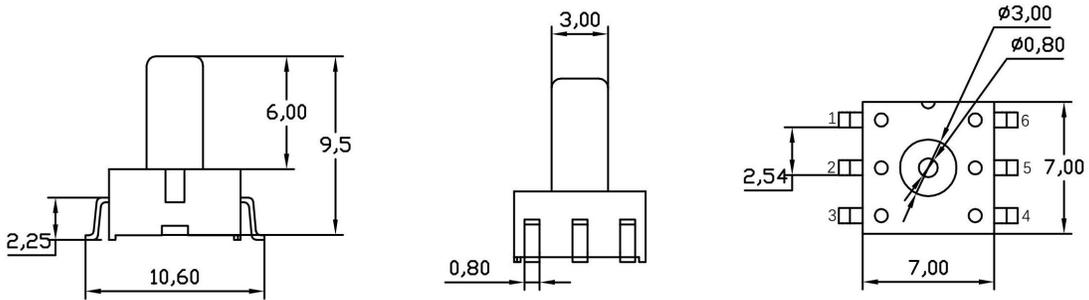


图 7

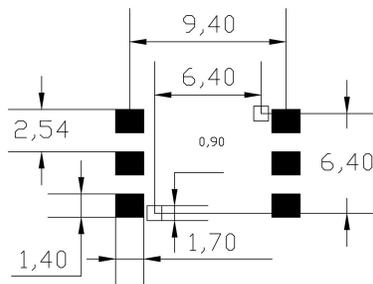


图 8. 推荐焊盘尺寸

9.选型指南

GZP 6859 D - 010KPP F01

GZP	芯感智
6859	产品大类
D	输出类型 A: 模拟输出 D: IIC 输出
010KPP	压力量程: 010 表示所测压力数值 (含 0~10, -10~0, -10~10) 压力单位: KP: KPa MP:MPa PS: PSI BA:Bar 压力类型: P: 正压(如 0~10) N: 负压 (如-10~0) W: 负压到正压(如-10~10) 故 010KPP 表示 0KPA 到 10KPA 的测量压力
F01 (附属项)	包装方式 B01: 编带 F01: 料管
XXX (如有)	定制项

表 9 选型指南

10.常用量程选型:

压力量程	型号
0 ~ 4kPa	GZP6859D004KPP F01
0 ~ 5kPa	GZP6859D005KPP F01 (B01 可选)
0 ~ 10kPa	GZP6859D010KPP F01
0 ~ 20kPa	GZP6859D020KPP F01
0 ~ 40kPa	GZP6859D040KPP F01
0 ~ 200kPa	GZP6859D201KPP F01
0 ~ 400kPa	GZP6859D401KPP F01
0 ~ 700kPa	GZP6859D701KPP F01
-1 ~ 1kPa	GZP6859D001KPW B01
-5 ~ 5kPa	GZP6859D005KPW F01
-20 ~ 20kPa	GZP6859D020KPW F01
-100 ~ 0kPa	GZP6859D101KPN F01
-100 ~ 100kPa	GZP6859D101KPW B01
更多定制量程及特殊参数料号, 请咨询制造商	

表 10. 常用量程表

11.使用注意事项

11.1 焊接

由于本产品为热容量较小的小型构造，因此请尽量减少来自外部的热量的影响。否则可能会因热变形而造成破损，引起特性变动。请使用非腐蚀性的松香型助焊剂。另外，由于产品暴露在外，因此请注意不要使助焊剂侵入内部。

1) 手焊接

- 请使用头部温度在 260 ~ 300 °C (30 W) 的电烙铁 在 5 秒以内实施作业。
- 在端子上施加负载进行焊接的情况下，由于输出可能会发生变化，因此请注意。
- 请保持电烙铁头洁净。

2) 回流焊接 (SMD 端子型)

推荐的回流炉温度设置条件如下所示

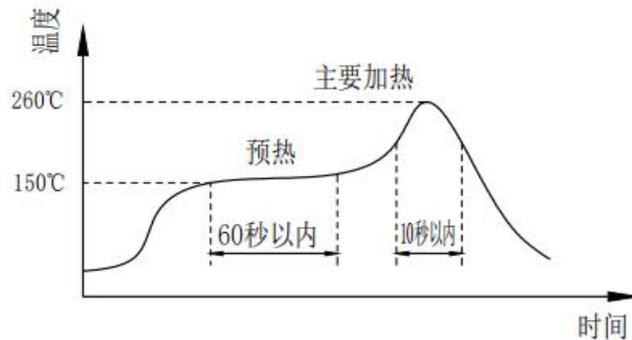


图 9. 回炉温度设置条件

- 3) 印刷板的翘度相对于整个传感器应保持在 0.05mm 以下，请对此进行管理。
- 4) 安装传感器后，对基板进行切割弯折时，请注意不要使焊接部产生应力。
- 5) 由于传感器的端子为外露构造，因此金属片等触摸端子后，会引发输出异常。请注意不要用金属片或者手等触摸。
- 6) 焊接后，为了防止基板的绝缘恶化而实施涂层时，请注意不要使传感器上面附着药剂。

11.2 清洗要求

- 1) 由于产品为开放型，因此请注意不要使清洗液侵入内部。
- 2) 使用超声波进行清洗时，可能会使产品发生故障，因此请避免使用超声波进行清洗。

11.3 存储与运输

- 1) 本产品为非防滴构造，因此请勿在可能溅到水等的场所中使用。
- 2) 请勿在产生凝露的环境中使用。另外，附着在传感器芯片上的水分冻结后，可能会造成传感器输出的变动或者破坏。
- 3) 压力传感器的芯片在构造上接触到光后，输出会发生变动。尤其是通过透明套等施加压力时，请避免使光接触到传感器的芯片。
- 4) 正常包装的压力传感器可通过普通输送工具运输。请注意：产品在运输过程中防止潮湿、冲击、晒伤和压力。

11.4 其他使用注意事项

- 1) 安装方法错误时，会造成事故，因此请注意。
- 2) 请避免采用超声波等施加高频振动的使用方法。
- 3) 能够直接使用的压力媒介仅为空气和无腐蚀性气体，除此以外的媒介，尤其是在腐蚀性介质和含有异物的媒介中使用，会造成故障和破损，因此请避免在上述环境中使用。
- 4) 压力导入口的内部配置有压力传感器芯片。从压力导入口插入针等异物后，会造成芯片破损和导入口堵塞，因此请绝对避免上述操作。另外，使用时请避免堵塞大气导入口。
- 5) 关于使用压力，请在额定压力的范围内使用。在范围外使用时，会造成破损。
- 6) 由于可能因静电而造成破坏，因此使用时请注意以下事项。
请将桌子上的带电物，作业人员接地，以使周围的静电安全放电。
- 7) 如有疑问，敬请垂询。

■ 请在实际使用状态下进行确认

由于本规格为产品单体规格，为了提高实际使用时的可靠性，请确认实际使用状态下的性能和品质。

12. 包装信息

料管信息 (单位为毫米)

每管数量:58 PCS

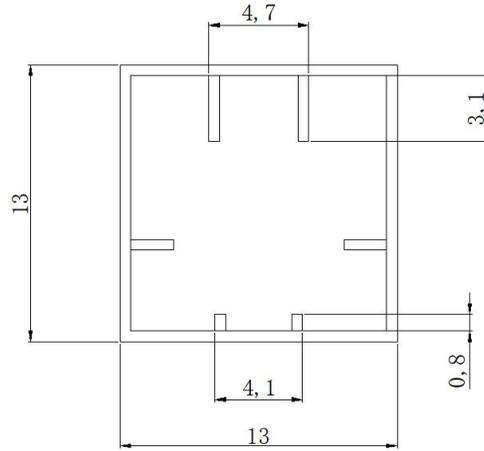


图 10.料管截面示意图

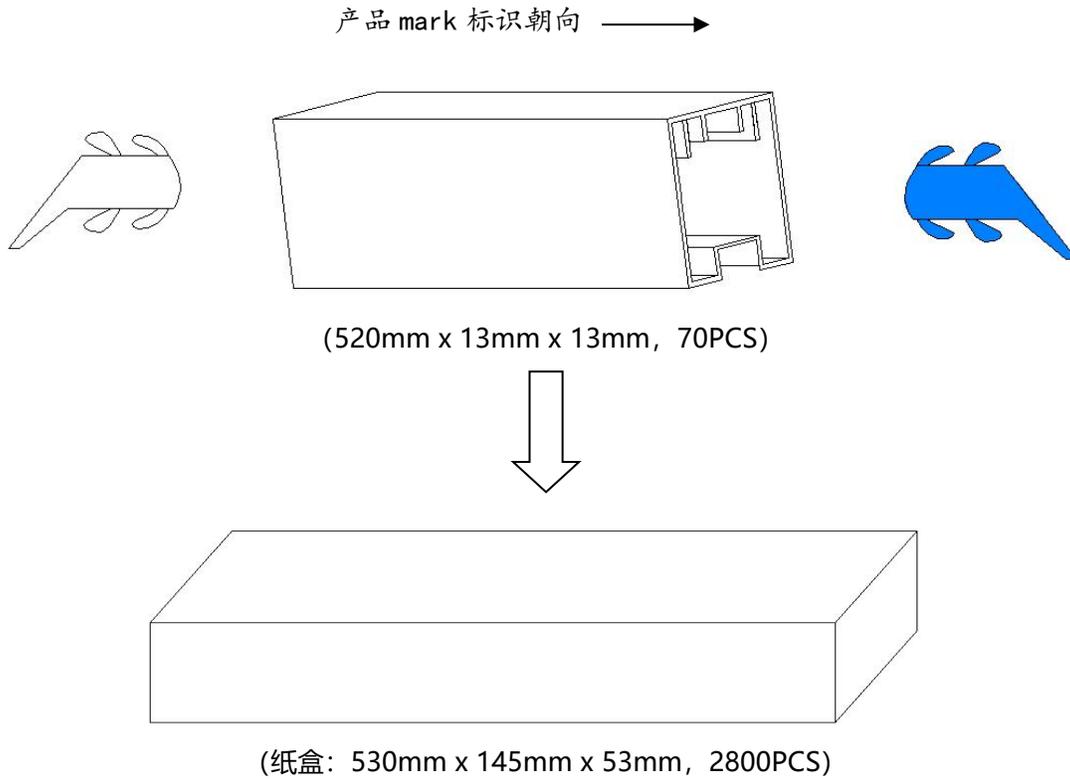


图 11.包装示意图

安全注意事项

本产品是使用一般电子设备用（通信设备，测量设备，工作机械等）的半导体部品而制成的。使用这些半导体部品的产品，可能会因外来干扰和浪涌而发生误动作和故障，因此请在实际使用状态下确认性能及品质。为以防万一，请在装置上进行安全设计（保险丝，断路器等保护电路的设置，装置多重化等），一旦发生误动作也不会侵害生命，身体，财产等。为防止受伤及事故的发生，请务必遵守以下事项：

·驱动电流和电压应在额定值以下使用。

·请按照电气定义进行接线。特别是对电源进行逆连接后，会因发热，冒烟，着火等电路损伤引发事故，因此敬请注意。

·对产品进行固定和对压力导入口进行连接时请慎重。

IIC Example Code (附件: IIC 代码案例)

```
#include <reg52.h>

#include <math.h>

#define DELAY_TIME 600

#define TRUE 1

#define FALSE 0

#define uchar unsigned char

#define uint unsigned int

//-----define IIC SCL,SDA port-----

sbit SCL = P1 ^ 7;

sbit SDA = P1 ^ 6;

//-----define Max7219 port-----

sbit Max7219_pinCLK = P2 ^ 2;

sbit Max7219_pinCS = P2 ^ 1;

sbit Max7219_pinDIN = P2 ^ 0;

//-----delay time_us-----

void DELAY(uint t)

{

    while (t != 0)

        t--;
```

```
}

//-----IIC START CONDITION-----

void I2C_Start(void)

{

    SDA = 1;          //SDA output high

    DELAY(DELAY_TIME);

    SCL = 1;

    DELAY(DELAY_TIME); //SCL output high

    SDA = 0;

    DELAY(DELAY_TIME);

    SCL = 0;

    DELAY(DELAY_TIME);

}

//-----IIC STOP CONDITION-----

void I2C_Stop(void)

{

    SDA = 0;          //SDA OUTPUT LOW

    DELAY(DELAY_TIME);

    SCL = 1;

    DELAY(DELAY_TIME);

    SDA = 1;

    DELAY(DELAY_TIME);

    SCL = 0;          //SCL OUTPUT LOW
```

```
        DELAY(DELAY_TIME);  
    }  
  
    //-----IIC SEND DATA "0"-----  
    void SEND_0(void)  
    {  
        SDA = 0;  
        DELAY(DELAY_TIME);  
        SCL = 1;  
        DELAY(DELAY_TIME);  
        SCL = 0;  
        DELAY(DELAY_TIME);  
    }  
  
    //-----IIC SEND DATA "1"-----  
    void SEND_1(void)  
    {  
        SDA = 1;  
        DELAY(DELAY_TIME);  
        SCL = 1;  
        DELAY(DELAY_TIME);  
        SCL = 0;  
        DELAY(DELAY_TIME);  
    }
```

```
//-----Check SLAVE's Acknowledge -----
```

```
bit Check_Acknowledge(void)
```

```
{
```

```
    SDA = 1;
```

```
    DELAY(DELAY_TIME);
```

```
    SCL = 1;
```

```
    DELAY(DELAY_TIME / 2);
```

```
    F0 = SDA;
```

```
    DELAY(DELAY_TIME / 2);
```

```
    SCL = 0;
```

```
    DELAY(DELAY_TIME);
```

```
    if (F0 == 1)
```

```
        return FALSE;
```

```
    return TRUE;
```

```
}
```

```
//-----Write One Byte of Data -----
```

```
void Writel2CByte(uchar b) reentrant
```

```
{
```

```
    char i;
```

```
    for (i = 0; i < 8; i++)
```

```
        if ((b << i) & 0x80)
```

```
        SEND_1());  
  
    else  
  
        SEND_0());  
  
    }  
  
//-----Read One Byte of Data -----  
  
uchar ReadI2CByte(void) reentrant  
{  
  
    char b = 0, i;  
    for (i = 0; i < 8; i++)  
    {  
  
        SDA = 1;  
  
        DELAY(DELAY_TIME);  
  
        SCL = 1;  
  
        DELAY(DELAY_TIME);  
  
        //DELAY(10);  
  
        F0 = SDA;  
  
        DELAY(DELAY_TIME);  
  
        //DELAY(10);  
  
        SCL = 0;  
  
        if (F0 == 1)  
        {  
  
            b = b << 1;  
  
            b = b | 0x01;  
  

```

```
    }  
    else  
        b = b << 1;  
    }  
    return b;  
}  
  
//-----write One Byte of Data,Data from MASTER to the SLAVER  
-----  
//-----SLAVER address bit:01101101-----  
void Write_One_Byte(uchar addr, uchar thedata) //Write "thedata" to the SLAVER's  
address of "addr"  
{  
    bit acktemp = 1;  
    I2C_Start();                //IIC START  
    WriteI2CByte(0xDA);        //IIC WRITE operation,SLAVER address  
bit:01101010  
    acktemp = Check_Acknowledge(); //check the SLAVER  
    WriteI2CByte(addr); /*address*/  
    acktemp = Check_Acknowledge();  
    WriteI2CByte(thedata); /*thedata*/  
    acktemp = Check_Acknowledge();  
    I2C_Stop();                //IIC STOP  
}
```

//-----Reaed One Byte of Data,Data from SLAVER to the MASTER

```
uchar Read_One_Byte(uchar addr)
```

```
{
```

```
    bit acktemp = 1;
```

```
    uchar mydata;
```

```
    I2C_Start();
```

```
    WriteI2CByte(0xDA);
```

```
    acktemp = Check_Acknowledge();
```

```
    WriteI2CByte(addr);
```

```
    acktemp = Check_Acknowledge();
```

```
    I2C_Start();
```

```
    WriteI2CByte(0xDB);           //IIC READ operation
```

```
    acktemp = Check_Acknowledge();
```

```
    mydata = ReadI2CByte();
```

```
    acktemp = Check_Acknowledge();
```

```
    I2C_Stop();
```

```
    return mydata;
```

```
}
```

```
//-----Delay_ms -----
```

```
void Delay_xms(uint x)
```

```
{
```

```
    uint i, j;
```

```
    for (i = 0; i < x; i++)
```

```
        for (j = 0; j < 112; j++)
            ;
    }

//-----Write One Byte to the Max7219-----
void Write_Max7219_byte(uchar DATA)
{
    uchar i;

    Max7219_pinCS = 0;    //CS low effect
    for (i = 8; i >= 1; i--)
    {
        Max7219_pinCLK = 0;

        Max7219_pinDIN = DATA & 0x80;

        DATA = DATA << 1;

        Max7219_pinCLK = 1;    //when pinCLK is high send the Data
    }
}

//-----decide which address shows the Data-----
void Write_Max7219(uchar address,uchar dat)
{
    Max7219_pinCS = 0;

    Write_Max7219_byte(address);

    Write_Max7219_byte(dat);

    Max7219_pinCS = 1;
}
```

```
//-----MAX_7219 Initialization-----  
  
void Init_MAX7219(void)  
{  
    Write_Max7219(0x09, 0xff); //译码方式: BCD 码  
    Write_Max7219(0x0a, 0x03); //亮度  
    Write_Max7219(0x0b, 0x07); //扫描界限: 8 个数码管显示  
    Write_Max7219(0x0c, 0x01); //掉电模式: 0, 普通模式: 1  
    Write_Max7219(0x0f, 0x01); //显示测试: 1; 测试结束, 正常显示: 0  
}  
  
void main(void)  
{  
    uchar yali1, yali2, yali3,wendu1,wendu2;  
    uchar temp_a5;  
    long int ad,temp;  
    long float pas;  
    uchar dis[8];  
    Init_MAX7219();  
    Delay_xms(1000);  
    Write_Max7219(0x0f, 0x00);  
    while (1)  
    {
```

```
temp_a5 = Read_One_Byte(0xA5); //Read ASIC Sys_config (读取系统配置值)

temp_a5 = temp_a5 & 0xFD; // (Raw_data_on: 0: output calibrated data,
输出为校准后的值, 即 0x06-0x0a 寄存器里的值为校准值)

Write_One_Byte(0xA5, temp_a5); //Set ADC output calibrated Data

Write_One_Byte(0x30, 0x0A); //indicate a combined conversion (once
temperature conversion immediately followed by once sensor signal conversion) (0x30 里
写入测量命令, 000: 单次温度测量; 001: 单次压力测量; 010: 组合: 单次压力和温度测量; 011:
休眠方式 (以一定的时间间隔执行组合模式测量) )

while ((Read_One_Byte(0x30) & 0x08) > 0); //Judge whether Data collection
is over 判断数据采集是否结束

// -----READ ADC output Data of Pressure -----

yali1 = Read_One_Byte(0x06);

yali2 = Read_One_Byte(0x07);

yali3 = Read_One_Byte(0x08);

ad = yali1 * 65536 + yali2 * 256 + yali3;

// -----READ ADC output Data of Temperature -----

wendu1= Read_One_Byte(0x09);

wendu2= Read_One_Byte(0x0a);

temp=wendu1*256+wendu2;

/*Conversion, the following is the conversion formula of 100kpa*/
```

```
if (ad > 8388608)          //超过 8388606 为负压值，需在显示终端做处理
{
    pas = (ad - 16777216)/64/1000;    //单位为 kpa
}
else
{
    pas = ad/64/1000;                //单位为 kpa
}
if (pas < 0)
    pas = fabs(pas);
/*Display program with Max7219*/
dis[0] = (long int)pas / 10000000;
dis[1] = (long int)pas % 10000000 / 1000000;
dis[2] = (long int)pas % 1000000 / 100000;
dis[3] = (long int)pas % 100000 / 10000;
dis[4] = (long int)pas % 10000 / 1000;
dis[5] = (long int)pas % 1000 / 100;
dis[6] = (long int)pas % 100 / 10;
dis[7] = (long int)pas % 10;
Write_Max7219(8, dis[0]);
Write_Max7219(7, dis[1]);
Write_Max7219(6, dis[2]);
Write_Max7219(5, dis[3]);
Write_Max7219(4, dis[4]);
Write_Max7219(3, dis[5]);
```

```
Write_Max7219(2, dis[6]);  
  
Write_Max7219(1, dis[7]);  
  
Delay_xms(100);           //delay 100ms  
  
}  
  
}
```