



GZP6829D

型压力传感器

数字输出
无铅产品

产品规格书

版本号： V1.3

文件发行日期： 2022.03.16



目录

1.产品特点	4
2.应用领域	4
3.概述	4
4.性能指标	4
5.电气特性	5
6.外形结构（单位为毫米）	6
7.电气连接	6
8.I ² C 通讯协议	7
9.寄存器描述	8
10.工作转换模式说明：	9
11.选型指南	10
12.常用量程	10
13.选型提示	10
14.使用注意事项	10
14.1.焊接	10
14.2.清洗要求	11
14.3.存储和运输	12
14.4.其他使用注意事项	12
15.包装信息	13
安全注意事项	14
IIC Example Code（附件：IIC 代码案例）	错误！未定义书签。
免责声明	21



文件修订历史

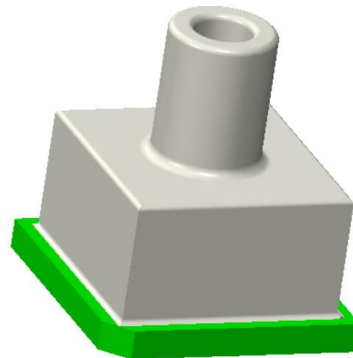
修订	描述	日期
V1.0	初始版本	2021.02.05
V1.1	增加电气参数 增加寄存器描述	2021.06.01
V1.2	完善产品信息，添加工作模式说明	2022.03.09
V1.3	调整产品归类	2022.03.16

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.产品特点

- 测量范围-100kPa…0kPa ~ 10kPa…200kPa
- 表压型
- 电源电压: 3.3V ~ 5.5V
- 适用于无腐蚀性的气体
- IIC 通讯



2.应用领域

- 电子血压计、呼吸机、制氧机、监护仪、真空拔罐等医疗保健领域
- 血压手表、血压手环等可穿戴设备
- 按摩器、按摩椅、气垫床等运动健身器材领域
- 真空泵、气泵、真空保鲜、净水机、压力仪表、气动元件等领域

3.概述

GZP6829D 型压力传感器采用 PCB 板封装形式，内有封装压力传感器与信号调理芯片，对传感器的偏移、灵敏度、温漂和非线性进行数字补偿。采用 21 位 ADC，并且调理芯片内置温度传感器，可以输出高精度的压力值和温度值。同时提供 IIC 通讯协议接口，抗干扰能力强。所有测量数据都经过充分校准和温度补偿。

GZP6829D 型压力传感器响应快、精度高，具有良好的线性以及长期稳定性。

GZP6829D 型压力传感器尺寸小、易安装，便于系统集成，广泛用于医疗电子、可穿戴设备、运动健身器材等领域。

4.性能指标

供电电源: $(5 \pm 0.25)V$ DC

参考温度: 25℃



表 1.性能指标

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	3× (量程≤60kPa)	Rated
	1.5× (量程>60kPa)	
破坏压力	4× (量程≤60kPa)	
	2× (量程>60kPa)	
补偿温度	0 ~ 60 (可定制)	°C
工作温度	-20 ~ 100	°C
贮存温度	-30 ~ 150	°C

* 精度为 0 ~ 70°C 范围内的输出误差，由压力的线性、重复性、迟滞组成，其压力量程不同，精度不同，请咨询客服获取更多细节。

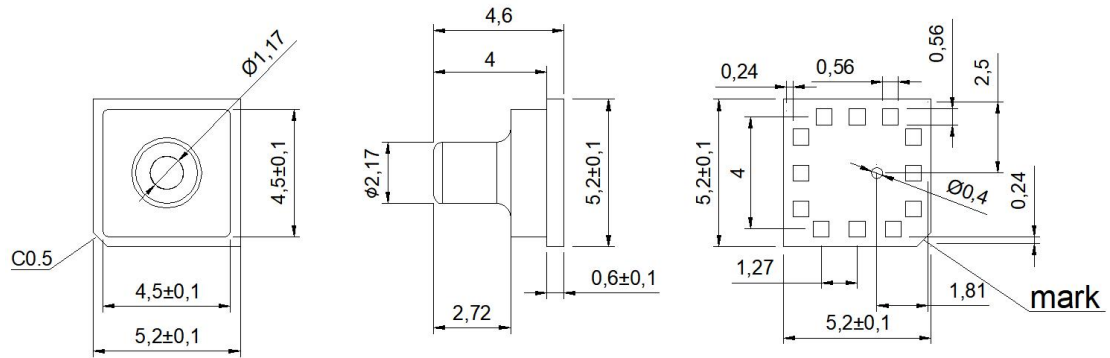
5.电气特性

表 2.电气特性

参数	最小值	典型值	最大	单位	备注
供电电压	3.3		5.5	V	
待机电流		100		nA	
电流消耗		5		uA	一次测量
LDO 输出		3.0		V	
PSRR		60		dB	
分辨率		24		Bits	
输出数据分辨率	21			Bits	LSB=(1/2 ²¹)*VDD
输入共模信号抑制比	80	110			
内置温度传感器准确度			±0.5	°C	@25°C
			±1	°C	-40 to 85 °C
温度分辨率	16			Bit	LSB = (1/256) °C
时钟脉冲频率			400	KHz	I2C 通讯



6.外形结构 (单位为毫米)

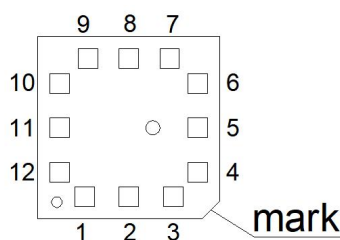


未注公差: $\pm 0.1\text{mm}$

图 1.外形结构

7.电气连接

表 3.引脚对应关系



引脚	名称	I/O	功能
1	NC	-	无定义
2	SDA	I/O	I ² C 双向数据线
3	SCL	I	I ² C 时钟线
4	NC	-	无定义
5	NC	-	无定义
6	NC	-	无定义
7	NC	-	无定义
8	NC	-	无定义
9	NC	-	无定义
10	GND	I	接地
11	NC	-	无定义
12	VDD	I	电源

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

8.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 2.2K）连接到 VDD，不通信时都保持为高电平。I²C 设备地址为 0x58。

■ I²C 时序图

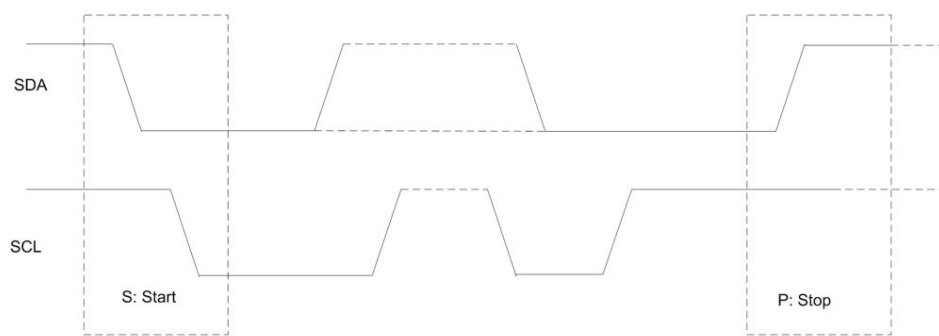
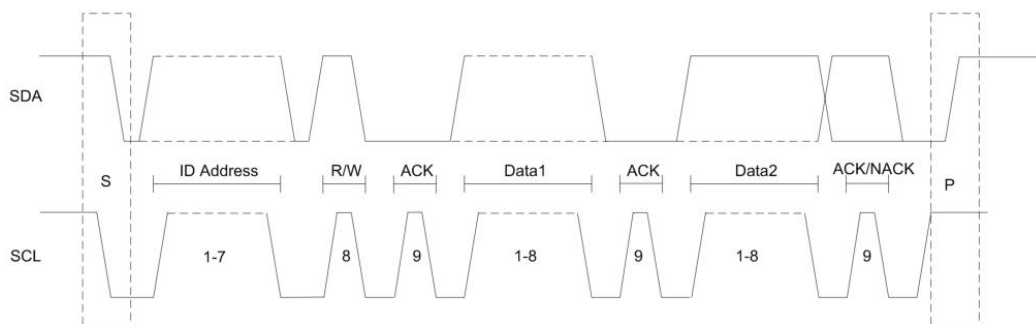


图 2.IIC 时序图

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





9.寄存器描述

表 4.寄存器描述

地址	描述	R/W	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	默认值
0x00	ID	R	ID<7:0>								0x58
0x01	Chip_Contr ol	R/W	reserved< 7:6>	data_rea dy	reserv ed	data_o ut	measure ment_ctrl	Active<1:0>		0x05	
0x02	CFG_OSR	R/W	OSR_T<7:5>			OSR_P<4:2>			MODE[1:0]		OTP
0x03	CFG_MEAS	R/W	reserved< 7:6>	T_SB[5:3]			PT_R[2:0]			OTP	
0x04	P_data	R	Data out<23:16>								0x00
0x05	P_data	R	Data out<15:8>								0x00
0x06	P_data	R	Data out<7:0>								0x00
0x07	T_data	R	Temp out<15:8>								0x00
0x08	T_data	R	Temp out<7:0>								0x00
0x24	CFG_OPER	R/W	reserved<7:1>						DAC_EN		OTP

Reg0x00: I2C 设备地址，默认地址为 0x58H。

Reg0x01: 芯片控制寄存器

Active<1:0>:00,芯片掉电; 01, 芯片启动;

measurement_ctrl: 0, 压力测量; 1, 温度测量;

data_out: 0, 输出校准数据; 1, 输出原始数据;

data_ready: 0, 数据转换未完成; 1, 数据转换完成。

Reg0x02

MODE[1:0]: 00: Sleep mode 睡眠模式, 01: Normal mode 正常模式, 10: One shot mode 单次采集模式, 11: Normal mode, cyclic measurement 正常模式循环测量

OSR_P[4:2] (压力过采样): 000: over sampling x 256

001: over sampling x 512

010: over sampling x 1024

011: over sampling x 2048

100: over sampling x 4096

101: over sampling x 8192

110: over sampling x 16384

111: over sampling x 32768

OSR_T[7:5] (温度过采样): 000: over sampling x 256

001: over sampling x 512

010: over sampling x 1024

011: over sampling x 2048

100: over sampling x 4096

101: over sampling x 8192

110: over sampling x 16384

111: over sampling x 32768



Reg0x03

PT_R[2:0]: 000: 64/1, 001: 32/1, 010: 16/1, 011: 8/1, 100: 4/1, 101: 1/1, Others: 128/1
(正常模式下, 压力/温度测量比)

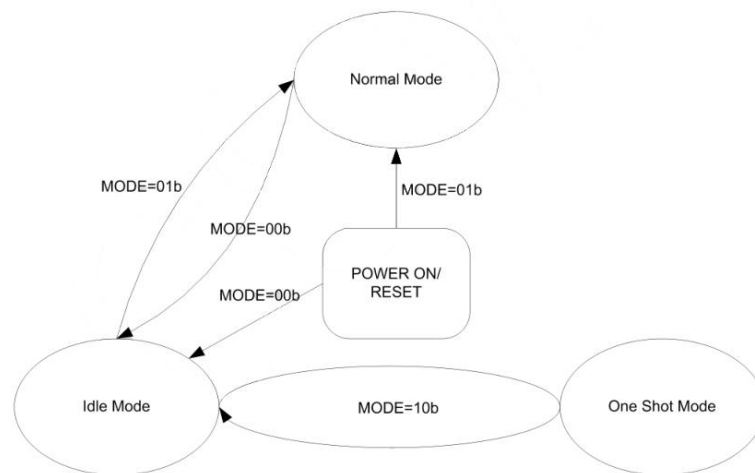
T_SB[5:3]: 000: 0ms, 001: 62.5ms, 010: 125ms, 011: 250ms, 100: 500ms, 101: 750ms,
110: 1000ms, 111: 2000ms (正常模式下的待机时间设置)

Reg0x04-Reg0x06 压力数据寄存器

Reg0x07-Reg0x08 温度数据寄存器

Reg0x24 DAC_EN: 0:禁用 DAC, 1:启用 DAC

10.工作转换模式说明:



■ 正常模式, 读取数据按照如下指令顺序进行操作:

1、VDD 上电

2、先往 0x01 地址里写 0x01, 启动芯片

3、延时 20ms 后续就可以从 0x04 连续读 5 个 bytes(会自动刷新数据)

4、前 3 个 bytes 为气压数据, 需要算补码, 具体为:

$sum = (0x04 \text{ 值} \times 2^{16} + 0x05 \text{ 值} \times 2^8 + 0x06 \text{ 值}),$

若 $sum < 8388608$, 则 $P = sum / 2^{21} \times 40 \times 1000$ (单位为 pa), (40kpa 量程计算公式)

若 $sum \geq 8388608$, $P = (sum - 16777216) / 2^{21} \times 40 \times 1000$ (单位为 pa) (40kpa 量程计算公式)



11.选型指南

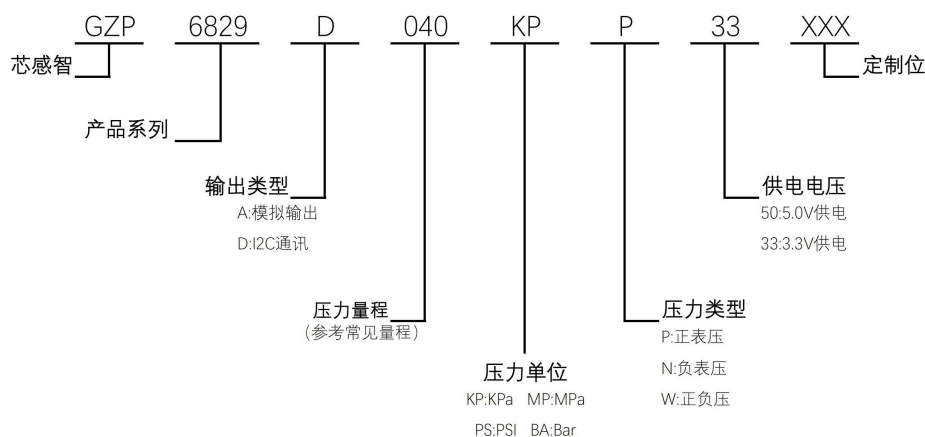


图 3.选型指南

12.常用量程

表 4. 常用量程表

压力量程 (kPa)	型号
0 ~ 10	GZP6829D010KPP50
0 ~ 40	GZP6829D040KPP50
0 ~ 100	GZP6829D100KPP50
-100 ~ 0	GZP6829D100KPN50
-100 ~ 100	GZP6829D100KPW50
更多定制量程及参数,请咨询我司客服或代理商	

13.选型提示

- 1.选型时请注意被测介质要与产品与介质相接触的部分相兼容。
- 2.若对产品的性能参数和功能上有特殊要求, 请与本公司商洽。

14.使用注意事项

14.1.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) DIP 焊接 (DIP 端子型)

- 在温度为 260 °C 以下的 DIP 焊锡槽内在 5 秒以内实施作业。
- 安装在热容量较小的基板上时, 由于可能会发生热变形, 因此请避免采用 DIP 焊接。

3) 回流焊接 (SMD 端子型)

推荐的回流炉温度设置条件如下所示

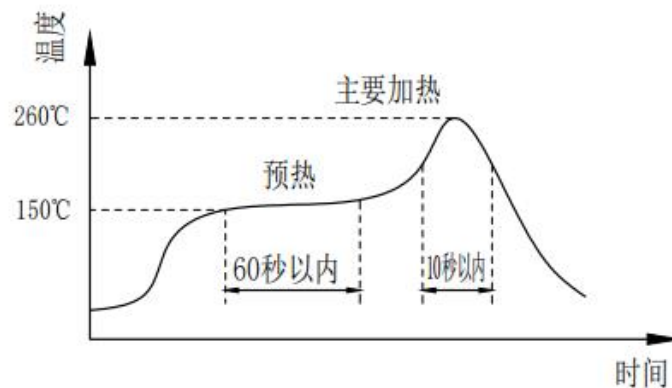


图 4.回流焊接

- 印刷电路板的走线请参照印刷电路板推荐规格图。
- 由于无法做到自校准, 因此请慎重地对准端子与走线的位置。
- 设置的温度为端子附近的印刷电路板上所测得的值。
- 因为由于装置, 条件等原因, 压力导入口的先端因为高温会发生溶解和变形, 务必请在实际的贴装条件下, 进行确认测试。

4) 焊接部的修正

- 请一次性完成修正。
- 对搭焊进行修正时, 请使用头部形状较平滑的电烙铁, 请勿追加涂敷助焊剂。
- 关于电烙铁头部的温度, 请使用在规格书所记载的温度以下的电烙铁。

5) 在端子上施加过度的力后, 会引发变形, 损害焊接性, 因此请避免使产品掉落, 或进行繁杂的使用。

6) 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。

7) 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。

8) 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。

9) 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。

14.2.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。



14.3.存储和运输

- 1) 本产品并非防滴构造，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。

14.4.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干燥空气。除此以外的媒介，尤其是在腐蚀性气体（有机溶剂气体，亚硫酸气体，硫化氢气体等）和含有水分，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) 根据所使用的压力，请充分注意产品的固定和套管，导入管的固定及选择。另外，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。



15. 包装信息

载带

每卷数量:500 pcs

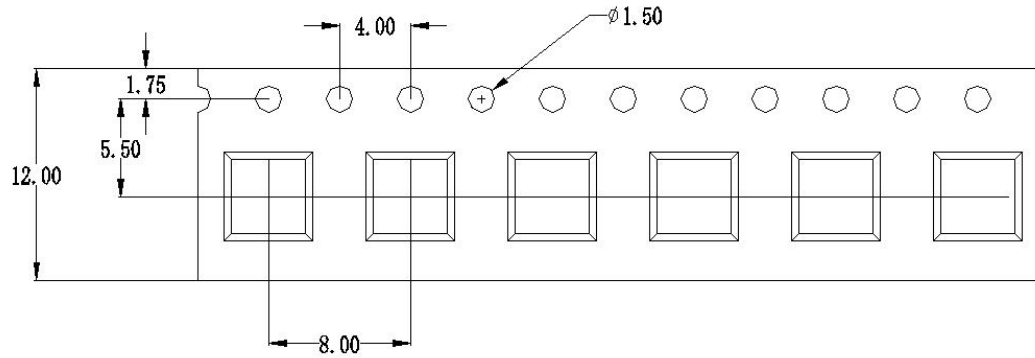


图 5. 载带

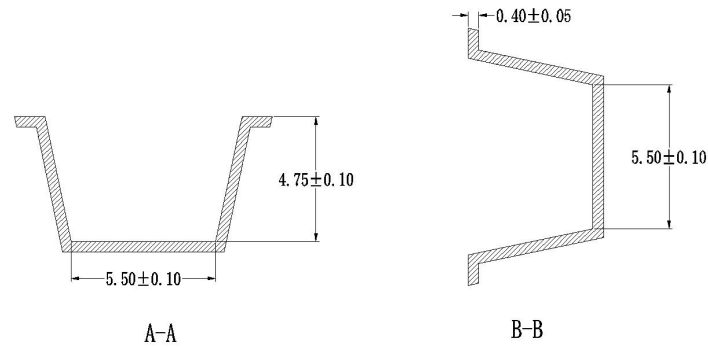


图 6. 载带口袋细节图

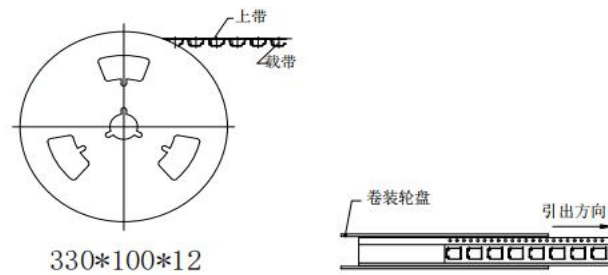


图 7. 载带转盘



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 (C51 Language)

```
#include <reg52.h>
#include <math.h>
#define DELAY_TIME 600    //Time-Delay Parameter 时延参数，可根据需要做适当调整
#define TRUE 1
#define FALSE 0
#define uchar unsigned char
#define uint unsigned int

float SPAN = 40; //SPAN is the span of the sensor 传感器的量程 0~40KPa

sbit SCL = P1 ^ 7; //IIC clock line 定义 IIC 总线时钟线
sbit SDA = P1 ^ 6; //IIC clock line 定义 IIC 总线时钟线

//Time-Delay Function 时延函数
void DELAY(uint t)
{
    while (t != 0)
        t--;
}

void I2C_Start(void) //IIC Start signal 发送 IIC 总线起始信号
{
    SDA = 1; //SDA output high SDA 输出高电平
    DELAY(DELAY_TIME);
    SCL = 1; //SCL output high SCL 输出高电平
    DELAY(DELAY_TIME);
    SDA = 0; //SDA output low SDA 输出低电平
    DELAY(DELAY_TIME);
    SCL = 0; //SCL output low SCL 输出低电平
    DELAY(DELAY_TIME);
}

void I2C_Stop(void) //IIC Stop signal 发送 IIC 总线停止信号
{
    SDA = 0;
    DELAY(DELAY_TIME);
    SCL = 1;
    DELAY(DELAY_TIME);
    SDA = 1;
    DELAY(DELAY_TIME);
}
```



```
SCL = 0;
DELAY(DELAY_TIME);
}

void SEND_0(void) //IIC send data "0" 向 IIC 总线发送"0"
{
    SDA = 0;
    DELAY(DELAY_TIME);
    SCL = 1;
    DELAY(DELAY_TIME);
    SCL = 0;
    DELAY(DELAY_TIME);
}

void SEND_1(void) //IIC send data "1" 向 IIC 总线发送"1"
{
    SDA = 1;
    DELAY(DELAY_TIME);
    SCL = 1;
    DELAY(DELAY_TIME);
    SCL = 0;
    DELAY(DELAY_TIME);
}

bit Check_Acknowledge(void) //Read ACK signal 读取 ACK 信号
{
    char F0 = 0;
    SDA = 1;
    DELAY(DELAY_TIME);
    SCL = 1;
    DELAY(DELAY_TIME / 2);
    F0 = SDA;
    DELAY(DELAY_TIME / 2);
    SCL = 0;
    DELAY(DELAY_TIME);
    if (F0)
        return FALSE;
    return TRUE;
}

void Writel2CByte(uchar b) reentrant //Write One Byte of Data 发送一个字节
{

```



```
char i;
for (i = 0; i < 8; i++)
    if ((b << i) & 0x80) //Send high bits first 先发送高位
        SEND_1();
    else
        SEND_0();
}

uchar ReadI2CByte(void) reentrant //Receive one byte 读取一个字节
{
    char b = 0, i, F0 = 0;
    for (i = 0; i < 8; i++)
    {
        SDA = 1;
        DELAY(DELAY_TIME);
        SCL = 1;
        DELAY(DELAY_TIME);
        F0 = SDA;
        DELAY(DELAY_TIME);
        SCL = 0;
        if (F0)
        {
            b = b << 1; //Receive high bits first 先读取高位
            b = b | 0x01;
        }
        else
            b = b << 1;
    }
    return b;
}

//Write a register's address and a command byte to the sensor
//向传感器写寄存器地址和一个命令字节
//"addr": register's address, "thedata": the command byte
void Write_One_Byte(uchar addr, uchar thedata)
{
    bit acktemp = 1;
    I2C_Start(); //IIC START Signal 发送 IIC 启动信号
    Writel2CByte(0x58 << 1 + 0); //The SLAVER address is 0x58
    //传感器的 IIC 总线地址为 0x58
    // The lowest bit of address is 0 means writing 地址值最低位为 0 表示写
    acktemp = Check_Acknowledge(); //check the SLAVER's ACK 检查传感器的 ACK
```



```

    Writel2CByte(addr); //Send the register's address 发送寄存器的地址值
    acktemp = Check_Acknowledge(); //check the SLAVER's ACK 检查传感器的 ACK
    Writel2CByte(thedata); //Write command to the sensor 向传感器写命令字节
    acktemp = Check_Acknowledge(); //check the SLAVER's ACK 检查传感器的 ACK
    I2C_Stop(); //IIC STOP Signal 发送 IIC 停止信号
}

//Read one byte of data from the sensor 从传感器读取一个字节
uchar Read_One_Byte(uchar addr)
{
    bit acktemp = 1;
    uchar mydata;
    I2C_Start(); //IIC START Signal 发送 IIC 启动信号
    Writel2CByte(0x58 << 1 + 0); //The SLAVER address is 0x58
    //传感器的 IIC 总线地址为 0x58
    // The lowest bit of address is 0 means writing 地址值最低位为 0 表示写
    acktemp = Check_Acknowledge(); //check the SLAVER's ACK 检查传感器的 ACK
    Writel2CByte(addr); //Send the register's address 发送寄存器的地址值
    acktemp = Check_Acknowledge(); //check the SLAVER's ACK 检查传感器的 ACK
    I2C_Start(); //IIC START Signal 发送 IIC 启动信号
    Writel2CByte(0x58 << 1 + 1); //The SLAVER address is 0x58
    //传感器的 IIC 总线地址为 0x58
    // The lowest bit of address is 1 means Reading 地址值最低位为 1 表示读
    acktemp = Check_Acknowledge(); //check the SLAVER's ACK 检查传感器的 ACK
    mydata = ReadI2CByte(); //Read the above register's data 读取上述寄存器的数据值
    acktemp = Check_Acknowledge(); //check the SLAVER's ACK 检查传感器的 ACK
    I2C_Stop(); //IIC STOP Signal 发送 IIC 停止信号
    return mydata;
}

//Ms Time-Delay Function Ms 延时函数
void Delay_xms(uint x)
{
    uint i, j;
    for (i = 0; i < x; i++)
        for (j = 0; j < 112; j++)
            ;
}

void main(void) //The main function 主函数
{
    uchar pressure_H, pressure_M, pressure_L, temperature_H, temperature_L;

```



```
//Temporary variables used to save bytes of pressure and temperature from the sensor
//临时变量，用于存放从传感器中读出的压力值和温度值的字节数据
long int pressure_ad,temperature_ad;
//Temporary variables used to save AD values of pressure and temperature from the
sensor
//临时变量，用于存放从传感器中读出的压力和温度的 AD 值
float pressure,temperature,Shift_N;
//pressure: actual pressure 变量 pressure 用于存放实际压力值
//temperature: actual temperature 变量 temperature 用于存放实际温度值
uchar byte1,byte2;
int EOFF;

Delay_xms(1000);
while (1)
{
    Write_One_Byte(0x01, 0x01);
    //Send 0x01 to the register whose address is 0x01 to start a data collection
    //向传感器 0x01 寄存器发送 0x01 以启动采集数据
    Delay_xms(20);

    pressure_H = Read_One_Byte(0x04); //Read bytes of pressure from the sensor
    pressure_M = Read_One_Byte(0x05); //从传感器中读出压力值的字节数据
    pressure_L = Read_One_Byte(0x06);
    pressure_ad = pressure_H * 65536 + pressure_M * 256 + pressure_L;
    //compute the AD pressure of the sensor 计算传感器 AD 转换后的压力值
    temperature_H = Read_One_Byte(0x07); //Read bytes of temperature from the sensor
    temperature_L = Read_One_Byte(0x08); //从传感器中读出温度值的字节数据
    temperature_ad = temperature_H * 256 + temperature_L;
    //compute the AD temperature of the sensor 计算传感器 AD 转换后的温度值

    //compute the actual pressure of the sensor 计算传感器实际的压力值
    //pressure's unit is Pa 变量 pressure 的单位为 Pa
    if (pressure_ad >= 8388608)
        pressure = (float) (pressure_ad - 16777216) / 2^21 * SPAN * 1000;
    else
        pressure = (float) pressure_ad / 2^21 * SPAN * 1000;

    //compute the actual temperature of the sensor 计算传感器实际的温度值
    //temperature's unit is Centigrade 变量 temperature 的单位为℃
    if(temperature_ad > 32768)
        temperature_ad -= 65536;
    byte1 = Read_One_Byte(0x20); //Read temperature parameter from the sensor
```



```
byte2 = Read_One_Byte(0x21); //Read temperature parameter from the sensor
if (byte1 == 0x0C) //According byte1 to evaluate the variable EOFF
    EOFF = 4096;
else if (byte1 == 0x8C) //根据 byte1 的值，求变量 EOFF 的值
    EOFF = -4096;
else if (byte1 == 0x0D)
    EOFF = 8192;
else if (byte1 == 0x8D)
    EOFF = -8192;
else if (byte1 == 0x0E)
    EOFF = 16384;
else if (byte1 == 0x8E)
    EOFF = -16384;
Shift_N = byte2 / 10; //compute the variable Shift_N 计算变量 Shift_N 的值
temperature = (temperature_ad - EOFF) / 2 ^ Shift_N + 25;
//the actual temperature of the sensor 传感器的实际温度值

printf("Actual pressure is %f Pa\r\n",pressure);
printf("Actual temperature is %f Centigrade\r\n\r\n",temperature);

Delay_xms(1000);
}
}
```



免责声明

本表中的信息已经过仔细审查，并被认为是准确的；但是，不对不准确之处承担任何责任。此外，此信息不会向此类设备的购买者传达制造商专利权下的任何许可。芯感智保留对此处的任何产品进行更改的权利，恕不另行通知。芯感智对其产品对任何特定用途的适用性不作任何保证、陈述或保证，也不承担因应用或使用任何产品或电路而产生的任何责任，并明确否认任何和所有责任，包括但不限于后果性或附带损害。典型参数可以而且确实在不同的应用中有所不同。客户的技术专家必须针对每个客户应用验证所有操作参数。