GZP6827D

Pressure Sensor

Digital Output(IIC)

Datasheet

Version: V1.8

Issued Data: 2023.11.17

Table of Contents

Document Revision History

Revision	Description	Date
V1.0	Initial version	2021.01.22
V1.1	Add electrical performance and pressure reading	2021.05.18
V1.2	Add cover, table of contents	2021.10.20
V1,3	Add working modes and instruction descriptions, adjust product classification	2022.03.14
V1.4	Modify packaging specifications	2022.08.15
V1.5	Modify I2C device address	2022.08.29
V1.6	Template modification	2023.05.12
V1.7	Update history figure library images	2023.09.27
V1.8	Update history figure library images	2023.11.07

1. Product Description

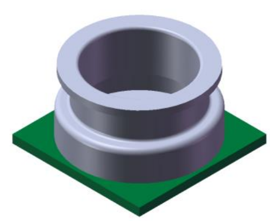
The GZP6827D type pressure sensor is packaged in a PCB board format, containing a sealed pressure sensor and a signal conditioning chip. It digitally compensates for the offset, sensitivity, temperature drift and non-linearity of the sensor. It uses a 21-bit ADC and the conditioning chip is equipped with a built-in temperature sensor to output high-precision pressure and temperature values. It also provides an IIC communication protocol interface, with strong anti-interference capability. All measurement data has been fully calibrated and temperature compensated.

The GZP6827D pressure sensor features fast response, high accuracy, excellent linearity, and long-term stability. It incorporates a digital filter with controllable signal bandwidth, supports periodic operation modes, and includes a 32-level FIFO (First In, First Out) function, making it adaptable to various application requirements while significantly reducing power consumption.

With its compact size and grooved pressure interface design, the GZP6827D pressure sensor facilitates easy assembly and sealing for users. It is particularly suitable for wearable devices with limited space.

1.1 Features

- Pressure range: 0 ~ 60PSI
- Gauge pressure type
- Suitable for non-corrosive gases
- Interior gel for moisture protection
- 1.8~3.6V power supply, low consumption
- Temperature compensated
- FIFO Function



1.2. Application Areas

- Electronic blood pressure monitors,electric breast pump.
- Wearable devices such as blood pressure watches and wristbands

2. Functional Description

This product is made with advanced micro-electromechanical principles. The crucial technology is the MEMS pressure sensor chip based on the silicon piezoresistive effect and the high-performance signal conditioning AISC chip. The silicon micro-piezoresistive MEMS pressure sensor chip is connected to a Wheatstone bridge composed of four strain-sensitive resistors. The output signal is amplified, temperature compensated and linearized by the ASIC chip. The linearity and temperature compensation of the transfer function are realized by the digital processing circuit in the ASIC. The high-precision pressure measurement in the full operating temperature range is achieved through the polynomial compensation algorithm and multi-point pressure calibration technology at multiple temperatures.

2.1 Pin Definition

The pin configuration of the pressure sensor is shown in Figure 1.

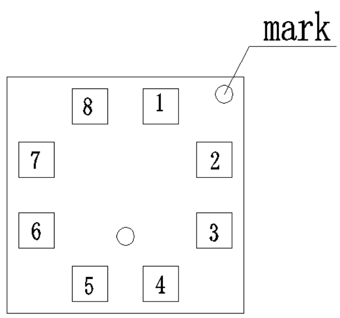


Fig.1 Pin Diagram

Tab.1 Pin Definition

Pin No.	Description	Remark
1	SCL	Signal output pin
2	SDA	Data output pin
3	INT	Interrupt output
4	NC	Floating pin
5	VDD	Power input positive
6	VDDIO	I/O Power input positive
7	GND	Power negative
8	GND	Power negative

2.2 Block Diagram

The sensor functional block diagram is shown in Figure 2.

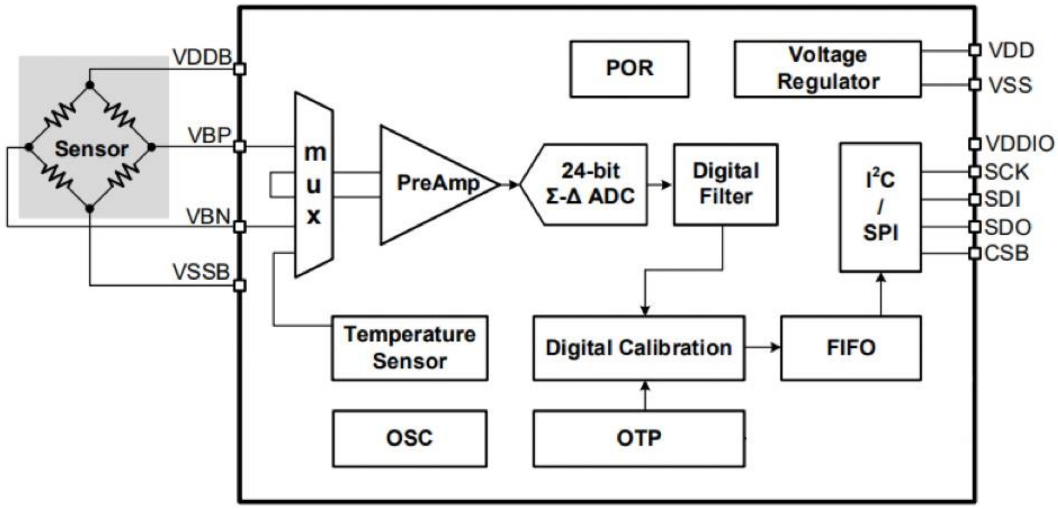


Fig.2 Block Diagram

2.3 Accuracy

The accuracy of the GZP6827D pressure sensor is composed of its linearity, repeatability, and hysteresis errors. The value calculated by the transfer function is the specified value of the sensor and also the theoretical value. The error of the sensor is equal to the difference between the actual output value of the sensor under the specified input pressure and the specified output value.

Overall Accuracy

The overall error includes more accuracy sources based on the product accuracy :

Pressure drift: The output deviation between the actual output voltage at zero point and full scale and the specified output voltage within the specified pressure range.

Temperature effect: The output deviation of zero point and full scale at different temperatures within the temperature range.

The overall accuracy is expressed by error band, and the data are shown in Figure 3 and Table 2 shown.

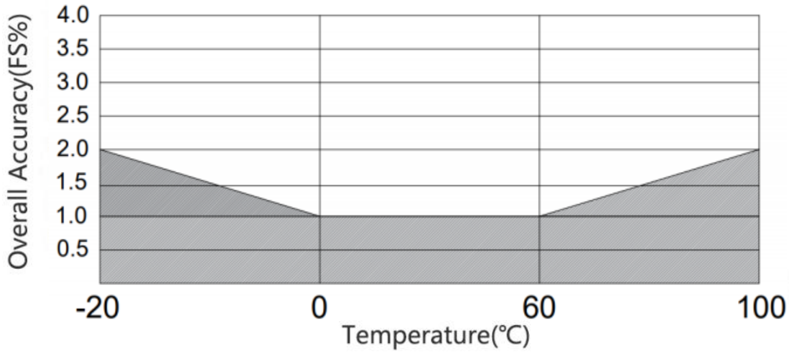


Fig.3 Relationship between overall accuracy and temperature

Tab.2 Overall Accuracy

Temperature (℃)	Overall Accuracy(Fs%)
-20~100	±2.0%
0~60	±1.0%

*Different pressure ranges have different overall errors. Please consult customer service for more details.

3. Technical Indicators

Measured at a power supply of 1.8~3.6V DC and a temperature of 25°C.

3.1 Maximum Rated Parameters

The maximum rated parameters of the sensor are shown in Table 3.

Tab.3 The maximum rated parameters

Parameter	Min.	Typical Value	Max.	Unit	Remark
Supply Voltage	-0.3		3.6	V	VDD/VDDIO
ESD Protection		±2		kV	HBM
Overload Pressure		1.5X		Rated	
Bursting Pressure		2X		Rated	
Working Temperature	-30		100		
Storage Temperature	-40		125		

1. Different pressure range may have different overload pressure and burst pressure, please consult Sencoch for more details.

2. Long exposure at the specified limits may cause degradation to the device.

3.2 Performance Indicators

The sensor performance indicators are shown in Table 4.

Tab.4 Sensor performance indicators

Parameter	Value	Unit	Remark
Pressure Range	-40...0 ~ 40...200	KPa	Customizable
Power Supply	1.8～3.6	V	
Standby Current	＜250	nA	25℃
Accuracy	±1	%Span	
Pressure Resolution	24	Bits	
Built-in Temperature Accuracy	±2	℃	@0~70℃
Compensation Temperature	0～60	℃	
Pull-up Resistors	4.7	K ohm	
Clock Frequency	400	KHz	Max.

* The different pressure range may have different accuracy, please consult Sencoch for more details.

4. Application Circuit

The recommended application circuit is shown in Figure 4.

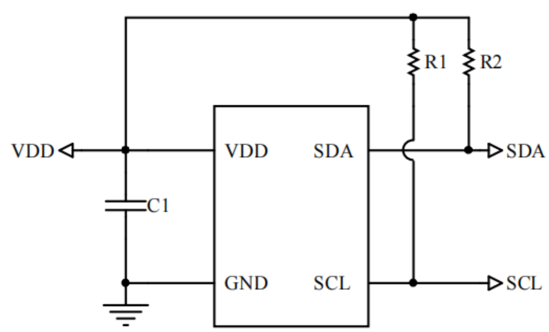


Fig.4 Application circuit

Notice :

1. Please confirm the electrical definition before soldering and assembly
2. Do not have any electrical connection with NC pin, otherwise it may cause sensor failure
3. Provide anti-static protection during soldering
4. Overload voltage (6.5Vdc) may burn out the circuit chip
5. Please add a 0.1uf capacitor between VDD and GND
6. Please pay attention to the power polarity during assembly

5. I²C Communication Protocol

The I²C bus uses SCL and SDA as signal lines. Both lines are connected to VDD through pull-up resistors (typical value 4.7K) and remain high when not communicating. The I²C device address is 0x38.

The I²C communication protocol has specific start (S) and stop (P) conditions. While SCL is high, a falling edge on SDA signals the start of data transmission. The I²C master device sequentially transmits the slave device's address (7 bits) and the read/write control bit. When the slave device recognizes this address, it generates an acknowledge signal and pulls SDA low in the ninth cycle. After receiving the slave device's acknowledgement, the master device continues to transmit the 8-bit register address and, upon receiving the acknowledgement, continues to send or read data. A rising edge on SDA while SCL is high signals the end of I²C communication. In addition to the start and stop signals, data transmitted by SDA must remain stable while SCL is high. The value transmitted by SDA can change while SCL is low. All data transmission in I²C communication is in 8-bit units, and an acknowledge signal is required after every 8 bits of data transmission to ensure continued transmission.

The I²C timing diagrams are shown in Figures 5 and 6.

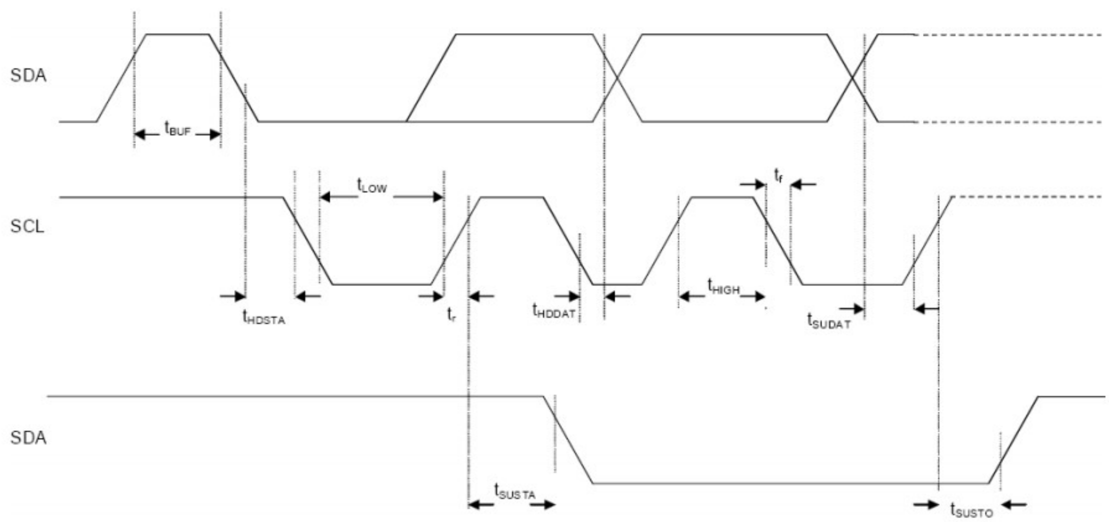


Fig.5. I²C Timing Diagram 1

6. Working Mode Description

The GZP6827D includes three working modes: "NOR", "CYC" and "CMD". The first two modes can be switched normally by the user. The "CMD" mode is a calibration mode before the product is manufactured and is not recommended for users to use.

Users can switch between the three working modes by sending control commands. If a mode conversion control command is sent while the product is in the process of measurement, this mode switching command will be delayed until the end of the current measurement cycle before being executed. Before this command is executed, any other commands sent subsequently will be ignored.

6.1 NOR Mode

This mode is the default operation mode after power-on. In this mode, the external controller (MCU) must actively send control commands to trigger a measurement process; otherwise, the product remains in a deep sleep state. Only after correctly receiving the control command will a measurement and calibration process for pressure and temperature begin, and then it will automatically enter a deep sleep state to save power until the next control command is received. During deep sleep, the current consumption is approximately 0.1μA. The NOR mode is suitable for applications with very low sampling rates or scenarios where it needs to work in synchronization with the main device.

6.2 CYC Mode

CYC mode (Cyclic mode) is an automatic periodic working mode. In this mode, the product will automatically switch between the measurement state and the sleep state periodically. Users can read the measurement results at any time.

In CYC mode, the product will automatically execute the operation corresponding to the last received control command until a new control command is received; if no control command is sent after entering CYC mode, the default measurement command will be executed.

The CYC mode is by default enabled with FIFO (First-In-First-Out). Through the digital interface, only the pressure value or temperature value can be read each time. Users can continuously read the measurement results saved in the FIFO until the FIFO is empty.

If FIFO is disabled, users must read 20-bit pressure values and 20-bit temperature values at once. When reading data, if the product is currently in the measurement process, the read data will be the previous measurement result. Based on the status byte read out, it is possible to determine whether the read measurement result has been updated.

6.3 I2C Testing Procedure

	From master to slave	S	Start	A	Acknowledge
	From slave to master	P	Stop	N	Not Acknowledge

6.3.1 Write Command

Write a single command

S	Device address	0	A	Command	A	P

or Write more commands

S	Device address	0	A	Command[15:8]	A	Command[7:0]	A	P

6.3.2 Read Command

(1) Read status word

S	Device address	1	A	Status	N	P

(2) When FIFO is enabled, read the measurement data in the FIFO.

S	Device address	1	A	Status	A	Data1[23:16]	A	Data2[15:8]	A	Data2[7:0]	N	P

(3) When FIFO is disabled, read 20-bit pressure and 20-bit temperature data from the output buffer.

S	Device address	1	A	Status	A	Data1 [23:16]	A	Data2 [15:8]	A	Data2 [7:0]	A	Data1 [23:16]	A	Data2 [15:8]	A	Data2 [7:0]	N	P

Note: After sending the write command, you need to wait for a period of time until the measurement by the device sensor is completed. Then, send the read command to read the measurement data.

6.4 Status Word

Regardless of which instruction is used, the valid data read out always starts from the status word (Status). The status word reflects the current state of the product after the latest command is executed. In the NOR mode, when no new command is executed, repeating the read operation will result in the same status word. In the CYC mode, since the chip operates automatically periodically, repeating the read operation may result in reading different status words. The specific definition of the status word can be found in Table 5.

Tab.5 The bit description of the status

Bits	**Significancy**	**Description**
Bit7	Flag bit	1: Measurement not completed 0: Measurement finished, data can be read
Bit6	Work mode	00: NOR mode 01: CYC mode 1x: CMD mode (for calibration purposes, not recommended for use)
Bit5	Reserved	Absolute value 0
Bit4	Data output	0: Output the original ADC data 1: Output the calibrated data
Bit3	FIFO Enable	0: FIFO function disabled 1: FIFO function enabled
Bit2	FIFO_FULL	0: FIFO data is not full 1: FIFO data is full
Bit1	FIFO_EMPTY	0: FIFO data is not empty 1: FIFO data is empty

6.5 INT Interrupt

After the measurement is initiated, INT becomes a low level "0"; after the measurement is completed, INT becomes a high level "1". After reading the data, the INT pin is automatically set to 0, and the INT pin becomes PIN 3.

6.6 Sleep mode

After the sensor completes the measurement, it enters a sleep standby state, with the standby power consumption being 0.1 uA.

6.7 IIC Common Instruction Descriptions

Tab.6 Common Instruction Descriptions

Command	**The setting or description for following (2 Bytes)**	**Return value**	**Description**
0xAC	(For oversampling rate settings, see Table 4)	20 bit of pressure data and 20 bit of temperature data	1:Measurement not completed 0:Measurement finished, data can be read
0xA8	——	——	Enter NOR mode
0xE0	——	——	Enter CYC mode
0xE1	(System configuration, see Table 5)	——	Modify the sleep time of the CYC mode
0xBA	——	——	Soft reset
0xBB	——	——	FIFO Reset

Tab.7 The oversampling rate setting and measurement time of ADC

Binary value	**OSR_P/OSR_T**	**Measurement time (ms)**
000	128x	33
001	64x	17
010	32x	8.4
011	16x	4.3
100	8x	2.2
101	4x	1.2
110	2x	0.7
111	1x	0.4

Tab.8 CYC System Configuration Description

Binary value	OSR_P/OSR_T	Measurement time (ms)
15	INT pin valid level	0: Low level valid 1: High level valid
14	FIFO data overflow interrupt	0: Disable full interrupt 1: Enable full interrupt
13	——	Reserve
12	FIFO Enable	0：Close FIFO 1:Open FIFO
11	Output data selection	0: Output the original ADC data 1: Output the calibrated data
[10：9]	——	Reserve
[8:6]	Temperature measurement rate control	000: Measure temperature every cycle 001: Measure temperature once every 2 cycles 010: Measure temperature once every 4 cycles 011: Measure temperature once every 8 cycles 100: Measure temperature once every 16 cycles 101: Measure temperature once every 32 cycles 110: Measure temperature once every 64 cycles 111: Measure temperature once every 128 cycles
[5:3]	Measurement cycle setting	000:5ms(200Hz) 001:10ms(100Hz) 010:20ms(50Hz) 011:40ms(25Hz) 100:80ms(12.5Hz) 101:250ms(4Hz) 110:500ms(2Hz) 111:1000ms(1Hz)
[2:0]	——	Reserve

6.8 0xAC Instruction Description

Instruction format is as follows:

Command byte	Data byte 0								Data byte 1							
--	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0xAC	0	OSR_T			0	OSR_P			0x00							

If a 32 times oversampling of pressure and temperature is set, the following command should be sent:

S	0x38	0	A	0xAC	A	0x22	A	0x00	A	P

7. Pressure Conversion Explanation

After reading the calibration data, a simple conversion needs to be performed on the unsigned number represented in AD value form.

To facilitate understanding, we assume the read calibration data to be: 0x08, 0x9B, 0xB0, 0xC5, 0x56, 0xAA, 0x00.

0x08 represents the status, and Bit7 being 0 indicates that data can be read. Assume that 0x9B, 0xB0, and 0xC5 are pressure data, and 0x56, 0xAA, and 0x00 are temperature data.

The conversion of pressure calibration values is as follows:

Convert 0x9B, 0xB0, and 0xC5 to decimal numbers as 10203333. In this calculation, it is assumed that the range used during calibration is 0Kpa - 40Kpa, and the corresponding AD output is 2516582 to 14260633 (15% AD to 85% AD).

The actual pressure value = (40 - 0) / (14260633 - 2516582) * (10203333 - 2516582) + 0 = 10.99 Kpa. The conversion of temperature calibration values is as follows:

Convert 0x56, 0xAA, and 0x00 to decimal numbers as 5679616. Since the calibration data read is in the form of a percentage, this percentage is numerically equal to the ratio of the decimal number we calculated to the maximum unsigned 24-bit number (16777216), so the percentage can be calculated as follows: 5679616 / 16777216 * 100% = 33.85%

The measurement range of the temperature sensor is specified as -50℃ to 150℃, so the calibration value = (150 - (-50)) * 33.85% - 50 = 17.70℃.

When using FIFO to read out 24-bit data, the high 20 bits can be used for the calculation in the above formula. The reference formula is as follows:

$$\text{Temperature}[^\circ\text{C}] = \frac{T[19:0]}{2^{20}} \cdot 200 - 50$$

Similarly, for the above pressure calculation formula, only the high 20 bits can be used for the calculation.

8. Structure Specification (unit: mm)

Refer to Figure 6 for sensor dimensions. (Tolerance ±0.5mm)

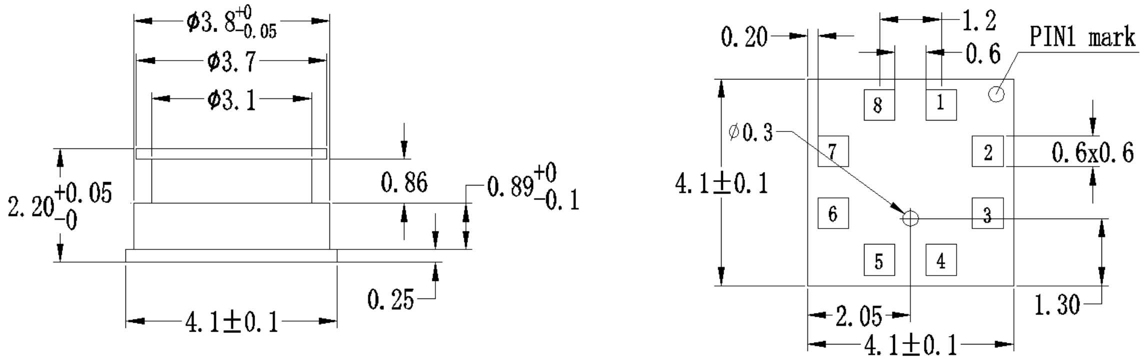


Fig.6 Sensor Dimensions

Recommended Footprint Layout refer to Figure 7.

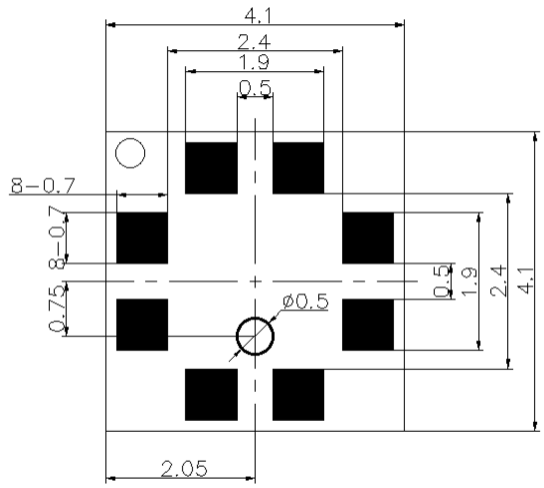


Fig.7 Recommended Footprint

9. Order Guide

GZP 6827 D 040KPP B01 WX

Tab.9 Order Guide

GZP	Pressure Sensor Series
6829	Product Series
D	Output type A: Analog output D: IIC output
C	IIC Communication Content Format
040KPP	Pressure range: 060 means the measured pressure value is 40 (including 0~40, -40~0, -40~40) Pressure unit: KP: KPa MP: MPa PS: PSI BA: Bar Pressure type: P: positive pressure (such as 0~40) N: negative pressure (such as -40~0) W: negative pressure to positive pressure (such as -40~40) Therefore, 040KPA means the measured pressure from 0 to 40KPA
B01	Packaging Method: B01: Tape
WX	Company interior code

10. Model Example

Tab.10 Model Example

Pressure Range	**Model**
0 ~ 40KPP	GZP6827D040KPP B01 WX

*For more customized ranges and special parameter part numbers, please consult the manufacturer.

11. Instructions for Use

11.1 Soldering

Since this product has a small structure with low heat capacity, please minimize the influence of heat from the outside. Otherwise, it may be damaged due to thermal deformation and cause changes in characteristics. Please use non-corrosive rosin type flux. In addition, since the product is exposed to the outside, please be careful not to allow flux to penetrate into the inside.

1) Manual soldering

· Please use a soldering iron with a head temperature of 260 to 300°C (30 W) and perform the work within 5 seconds.

· When soldering with a load applied to the terminals, the output may change, so be careful.

Please keep the soldering iron tip clean.

2) Reflow soldering (SMD terminal type)

The recommended reflow oven temperature setting conditions are show:

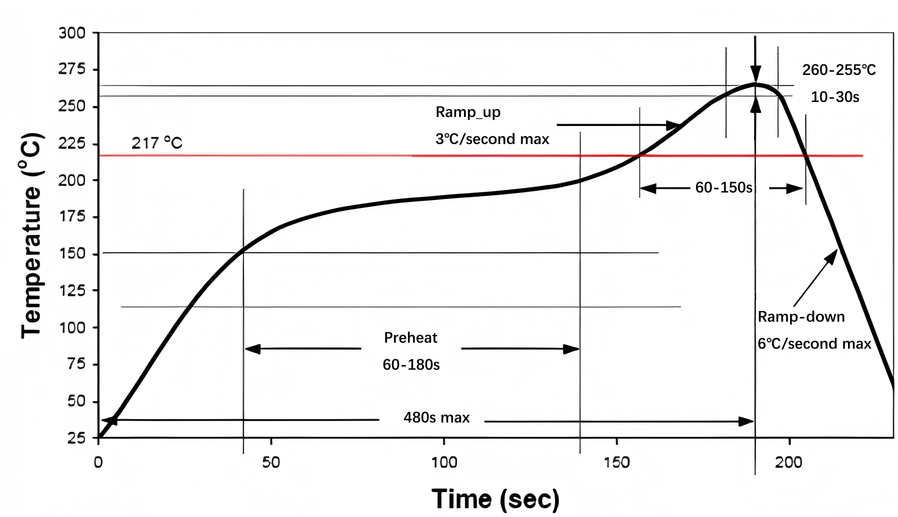


Fig.8 Remelting temperature setting conditions

3) The warping of the printed circuit board relative to the entire sensor should be kept below 0.05mm. Please manage this.

4) After installing the sensor, when cutting and bending the substrate, be careful not to cause stress on the soldered parts.

5) Since the sensor terminals are exposed, if metal pieces touch the terminals, abnormal output will occur. Be careful not to touch them with metal pieces or hands.

6) After soldering, when coating is applied to prevent insulation degradation of the substrate, be careful not to allow chemicals to adhere to the sensor.

11.2 Cleaning Requirements

1) Since the product is an open type, be careful not to allow cleaning fluid to enter the interior.

2) Please avoid using ultrasonic cleaning as it may cause product failure.

11.3 Storage and Transportation

1) This product is not drip-proof, so do not use it in a location where it may be splashed with water.

2) Do not use in an environment where condensation occurs. In addition, if moisture attached to the sensor chip freezes, it may cause changes in sensor output or damage.

3) The chip of the pressure sensor is structurally exposed to light, and the output will change. Especially when applying pressure through a transparent cover, please avoid light from reaching the chip of the sensor.

4) Normally packaged pressure sensors can be transported by ordinary transportation tools. Please note: The product should be protected from moisture, impact, sunburn and pressure during transportation.

11.4 Other Precautions

1) If the installation method is incorrect, it may cause an accident, so please be careful.

2) Avoid using the product in a manner that applies high-frequency vibrations, such as ultrasonic waves.

3) The only pressure media that can be used directly are air and non-corrosive gases. Other media, especially when used in corrosive media or media containing foreign matter, may cause malfunctions and damage, so please avoid using it in the above environment .

4) There is a pressure sensor chip inside the pressure inlet. If a needle or other foreign object is inserted into the pressure inlet, the chip will be damaged and the inlet will be blocked, so please avoid such operations. In addition, please avoid blocking the atmosphere inlet when using it.

5) Please use the pressure within the rated pressure range. Using outside the range may cause damage.

6) Since static electricity may cause damage, please pay attention to the following matters when using.

Please ground the charged objects and workers on the table to discharge the static electricity around safely.

7） If you have any questions, please feel free to ask.

12. Packing Information

Tape Packing:

2500PCS/Tape

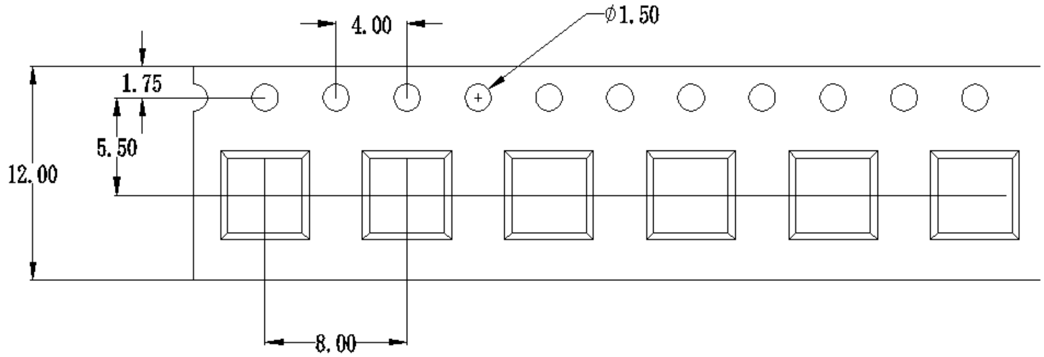


Fig.9 Tape

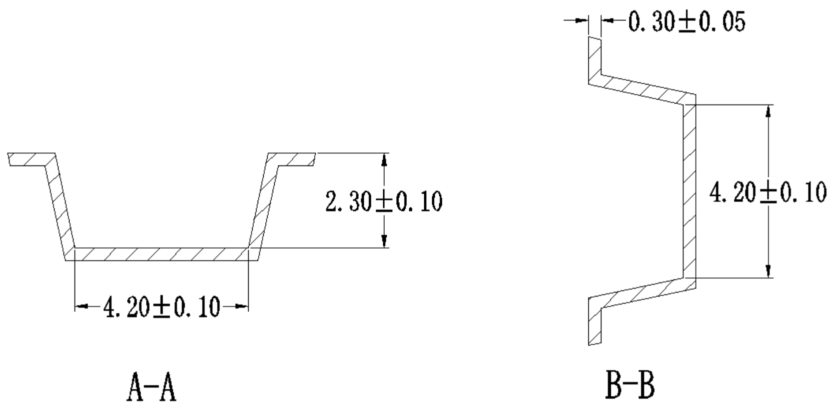


Fig.10 Detail of the tape

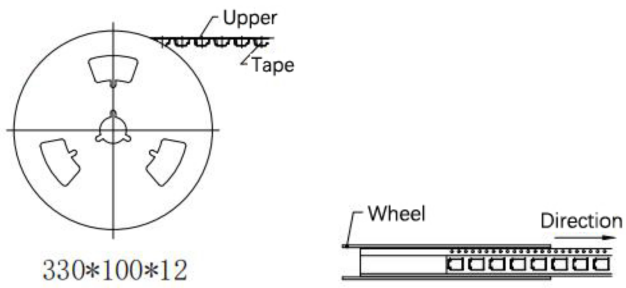


Fig.11 Tape Carrier

Safety Precautions

This product is made of semiconductor components for general electronic equipment (communication equipment, measuring equipment, working machinery, etc.). Products using these semiconductor components may malfunction and fail due to external interference and surges, so please confirm the performance and quality under actual use. To be on the safe side, please perform safety design on the device (setting of protection circuits such as fuses and circuit breakers, multiple devices, etc.) so that life, body, property, etc. will not be harmed in the event of a malfunction. To prevent injuries and accidents, please be sure to comply with the following matters:

·The driving current and voltage should be used below the rated values.

Please wire according to the electrical definition . In particular, reverse connection of the power supply may cause accidents due to circuit damage such as heat, smoke, and fire, so please be careful.

·Be careful when fixing the product and connecting the pressure inlet .

Warranty and Disclaimer

The information in this sheet has been carefully reviewed and is believed to be accurate; however, no responsibility is assumed for inaccuracies. Furthermore, this information does not convey to the purchaser of such devices any license under the patent rights to the manufacturer. Sencoch Technology reserves the right to make changes without further notice to any product herein. Sencoch Technology makes no warranty, representation or guarantee regarding the suitability of its product for any particular purpose, nor does Sencoch Technology assume any liability arising out of the application or use of any product or circuit and specifically disclaims any and all liability, including without limitation consequential or incidental damages. Typical parameters can and do vary in different applications. All operating parameters must be validated for each customer application by customer’s technical experts. Sencoch Technology does not convey any license under its patent rights nor the rights of others.

IIC Example Code (Attachment: IIC Code Example)

```
//**************************************************************************************************
//********************************************数码管显示压力及温度
//**********************************************STC12+MAX7219********************************
//*****************************CLK=P2^2 CS=P2^1
DIN=P2^0****************************************************************************************
//********************************************SCL=P1^7
SDA=P1^6**************************************************************************************
//**************************************************************************************************
#include <STC12C5A60S2.H>
#include <stdio.h>
#include <math.h>
#include "MAX7219.h"
#include "GZP6827D.h"
#include "IIC.h"


extern float pressure_kpa ;//, temp = 0.0;//float 4 字节
extern unsigned long pressure_pa ;
extern unsigned long temp ;
void Delay300ms()		//@11.0592MHz
{
	unsigned char i, j, k;


	i = 13;
	j = 156;
```

```
        k = 83;
        do
        {
                do
                {
                        while (--k);
                } while (--j);
        } while (--i);
}

void main()
{
                unsigned char dis[8] = {0,0,0,0,0,0,0,0};
                Delay_Ms(DELAY_TIME);
                Init_MAX7219();
        while(1)
        {
        GZP6827D_get_cal();
                dis[0] = (unsigned char)(pressure_pa / 10000000);
                        dis[1] = (unsigned char)(pressure_pa % 10000000 / 1000000);
                                dis[2] = (unsigned char)(pressure_pa % 1000000 / 100000);
                                        dis[3] = (unsigned char)(pressure_pa % 100000 / 10000);
                                                dis[4] = (unsigned char)(pressure_pa % 10000 / 1000);
                                                        dis[5] = (unsigned char)(pressure_pa % 1000 / 100);
                                                                dis[6] = (unsigned char)(pressure_pa % 100 / 10);
```

```
                                    dis[7] = (unsigned char)(pressure_pa % 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]|0x80);      //显示小数点
                            Write_Max7219(3, dis[5]);
                                Write_Max7219(2, dis[6]);
                                    Write_Max7219(1, dis[7]);
  Delay300ms();

         GZP6827D_get_cal();
         temp=temp*10;
         dis[0] = (unsigned char)(temp / 10000000);
         dis[1] = (unsigned char)(temp % 10000000 / 1000000);
             dis[2] = (unsigned char)(temp % 1000000 / 100000);
                 dis[3] = (unsigned char)(temp % 100000 / 10000);
                     dis[4] = (unsigned char)(temp % 10000 / 1000);
                         dis[5] = (unsigned char)(temp % 1000 / 100);
                             dis[6] = (unsigned char)(temp % 100 / 10);
                                 dis[7] = (unsigned char)(temp % 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]);        //显示小数点
                                                Write_Max7219(3, dis[5]);
                                                      Write_Max7219(2, dis[6]|0x80);
                                                            Write_Max7219(1, dis[7]);
                    Delay300ms();
      }
}

#include "GZP6827D.h"
#include <math.h>
// Define the upper and lower limits of the calibration pressure
#define PMIN 0   //Zero range pressure for example 0Kpa
#define PMAX 40   //Full Point Pressure Value, for example 40Kpa
#define DMIN 104,857.6   //AD value corresponding to pressure zero, for example
10%AD=2^20*0.1
#define DMAX   943,718.4 //AD Value Corresponding to Full Pressure Range, for example
90%AD=2^20*0.9

float pressure_kpa = 0.0;//, temp = 0.0;
unsigned long pressure_pa = 0;
unsigned long temp = 0.0;
```

```
//The 7-bit IIC address of the JHM1400 is 0x38

unsigned char Device_Address = 0x38 << 1;


//Read the status of IIC and judge whether IIC is busy

unsigned char GZP6827D_IsBusy(void)

{

    unsigned char status;

    GZP6827D_IIC_Read(Device_Address, &status, 1);

    status = (status >> 7) & 0x01;

    return status;

}


void GZP6827D_get_cal(void)

{

    unsigned char buffer[6] = {0};

    unsigned long Dtest = 0;

    unsigned int temp_raw = 0;

    //Send 0xAC command and read the returned six-byte data

    buffer[0] = 0xAC;

    GZP6827D_IIC_Write(Device_Address, buffer, 1);

    Delay_Ms(DELAY_TIME);

    while (1)

    {

        if (GZP6827D_IsBusy())

        {
```

```
            Delay_Ms(DELAY_TIME);
        }
        else
            break;
    }
    GZP6827D_IIC_Read(Device_Address, buffer, 6);

    //The returned pressure and temperature values are converted into actual values according to the calibration range
    Dtest = (unsigned long)((((unsigned long)buffer[1]) << 12) | (((unsigned int)buffer[2]) << 4) | ((unsigned char)buffer[3]>>4));
    temp_raw =((((unsigned char)buffer[3]&0x0f)<< 16)) |((unsigned int)buffer[4] << 8) | (buffer[5] ));
    if (Dtest != 0)
    {
        pressure_kpa = (float) ((PMAX-PMIN)/(DMAX-DMIN)*(Dtest-DMIN)+PMIN); //单位：KPa
        pressure_pa = (unsigned long) (pressure_kpa * 1000.0);    //单位：Pa
    }
    else
    {
        pressure_kpa = 0.0;    //单位：KPa
        pressure_pa = 0;    //单位：Pa
    }
    temp = (double)temp_raw /1048576   * 200 - 50;
```

```
}

//Write a byte of data through IIC
unsigned char GZP6827D_IIC_Write(unsigned char address, unsigned char *buf, unsigned
char count)
{
    unsigned char timeout, ack;
    address &= 0xFE;
    Start();
    Delay_Ms(DELAY_TIME);
    SendByte(address);
    /*  Set_SDA_INPUT();*/
    Delay_Ms(DELAY_TIME);
    timeout = 0;
    do
    {
        ack = Check_ACK();
        timeout++;
        if (timeout == 10)///////////////
        {
            Stop();
            return 1;
        }
    } while (ack);
    while (count)
```

```
    {
        SendByte(*buf);
        /*  Set_SDA_INPUT();*/
        Delay_Ms(DELAY_TIME);
        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
unsigned char GZP6827D_IIC_Read(unsigned char address, unsigned char *buf, unsigned
char count)
{
```

```
unsigned char timeout, ack;
address |= 0x01;
Start();
SendByte(address);
/*    Set_SDA_INPUT();*/
Delay_Ms(DELAY_TIME);
timeout = 0;
do
{
      ack = Check_ACK();
      timeout++;
      if (timeout == 10)
      {
            Stop();
            return 1;
      }
} while (ack);
Delay_Ms(DELAY_TIME);
while (count)
{
      *buf = ReceiveByte();
      if (count != 1)
            Send_ACK();
      buf++;
      count--;
```

```
    }
    Stop();
    return 0;
}

#include "IIC.h"
//****************************************************
//MS 延时函数(12M 晶振下测试)
//****************************************************
void Delay_Ms(unsigned char n)
{
    unsigned char i,j;      //char 改成 int
        for(i=0;i<n;i++)
            for(j=0;j<123;j++);
}

//Start signal
void Start(void)
{
    /*  Set_SDA_OUTPUT();   */
    SDA = 1;
        Delay_Ms(DELAY_TIME);
            SCL = 1;
                Delay_Ms(DELAY_TIME);
                    SDA = 0;
```

```
                    Delay_Ms(DELAY_TIME);
                        SCL = 0;
                            Delay_Ms(DELAY_TIME);//**
}

//Stop signal
void Stop(void)
{
    /*  Set_SDA_OUTPUT();  */
    SDA = 0;
        Delay_Ms(DELAY_TIME);
            SCL = 1;
                Delay_Ms(DELAY_TIME);
                    SDA = 1;
                        Delay_Ms(DELAY_TIME);
                            SCL = 0;               //**
                                Delay_Ms(DELAY_TIME); //**
}

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

```
            Delay_Ms(DELAY_TIME); //**
                SCL = 1;
                    Delay_Ms(DELAY_TIME / 2);
                        ack = SDA;
                            Delay_Ms(DELAY_TIME / 2);
                                SCL = 0;
                                    Delay_Ms(DELAY_TIME);//**
    /*  Set_SDA_OUTPUT();   */
                                        return ack;
    //Delay_Ms(DELAY_TIME);//**
}

//Send ACK signal
void Send_ACK(void)
{
    /*  Set_SDA_OUTPUT();   */
        SDA = 0;
            Delay_Ms(DELAY_TIME);
                SCL = 1;
                    Delay_Ms(DELAY_TIME);
                        SCL = 0;
                            Delay_Ms(DELAY_TIME);
                                SDA = 1;
                                    Delay_Ms(DELAY_TIME);
}
```

```
//Send one byte
void SendByte(unsigned char byte1)
{
    unsigned char i = 0;
    /*  Set_SDA_OUTPUT();   */
        do
            {
                    if (byte1 & 0x80)
                {
                        SDA = 1;
                        }
                            else
                                {
                                    SDA = 0;
                                        }
                        Delay_Ms(DELAY_TIME);
                            SCL = 1;
                                Delay_Ms(DELAY_TIME);
                                    byte1 <<= 1;
                                        i++;
                                            SCL = 0;
        //Delay_Ms(DELAY_TIME);//**
            } while (i < 8);
                        SCL = 0;
```

```
Delay_Ms(DELAY_TIME);
}

//Receive one byte
unsigned char ReceiveByte(void)
{
    unsigned char i = 0, tmp = 0;
    /*  Set_SDA_INPUT();*/
        do
            {
                tmp <<= 1;
                    SCL = 1;
                        Delay_Ms(DELAY_TIME);
                            if (SDA)
                                {
                                    tmp |= 1;
                                    }
                                        SCL = 0;
                                            Delay_Ms(DELAY_TIME);
                                                i++;
            } while (i < 8);
                    return tmp;
}
```

```
/******数码管驱动程序************/
/******管脚配置
            CLK=P2^2
             CS=P2^1
             DIN=P2^0
            SCL=P1^7
             SDA=P1^6
             *************************/
#include "MAX7219.h"
#include <STC12C5A60S2.H>

#define uchar unsigned char

sbit Max7219_CLK=P2^2;
sbit Max7219_CS=P2^1;
sbit Max7219_DIN=P2^0;

//-------------Write One Byte to the Max7219-------------
void Write_Max7219_byte(uchar Data)
{
    unsigned char i;
    Max7219_CS = 0;          //CS low effect
    for (i = 8; i >= 1; i--)
    {
```

```
        Max7219_CLK = 0;
        Max7219_DIN = Data & 0x80;
        Data = Data << 1;
        Max7219_CLK = 1;                //when pinCLK is high send the Data
    }
}

//-------------decide which address shows the Data-------------
void Write_Max7219(uchar address,uchar dat)
{
    Max7219_CS = 0;
    Write_Max7219_byte(address);
    Write_Max7219_byte(dat);
    Max7219_CS = 1;
}

//-------------MAX_7219 Initialization-------------
void Init_MAX7219(void)
{
    Write_Max7219(0x09, 0xff);    //译码方式：BCD 码
    Write_Max7219(0x0a, 0x01);    //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, 0x00);    //显示测试：1；测试结束，正常显示：0
}
```