# GZP6869DC

## Pressure Sensor

**Digital Output (IIC)**

Datasheet

Version: V1.4

Issued Date: 2026.02.11

# Table of Contents

1 Product Description ..... 4
1.1 Features ..... 4
1.2 Applications ..... 4
2 Function Description ..... 5
2.1 Block Diagram ..... 5
2.2 Pin Definition ..... 错误!未定义书签。
2.3 Accuracy ..... 6
3 Technical Indicators ..... 7
3.1 Maximum Ratings ..... 7
3.2 Performance Indicators ..... 8
4 Application Circuit ..... 9
5 $I^2C$ Communication Protocol ..... 9
6 Register Description ..... 10
7 Working Mode Description ..... 12
8 Structure Specification (unit:mm) ..... 13
9 Order Guide ..... 15
10 Model Example ..... 15
11 Instruction for Use ..... 16
11.1 Soldering ..... 16
11.2 Cleaning Requirements ..... 17
11.3 Storage and Transportation ..... 17
11.4 Other Precautions ..... 17
12 Packaging Information ..... 18
Safety Precautions ..... 20
IIC Example Code (Attachment: IIC Code Example) ..... 21

## Document Revision History

| Revision | Description | Date |
|---|---|---|
| V1.0 | Initial version | 2025.01.11 |
| V1.1 | Update drawings and templates | 2025.03.19 |
| V1.2 | Update code | 2025.07.05 |
| V1.3 | Update product features | 2025.08.01 |
| V1.4 | Add product house(L2) | 2026.02.11 |

The company reserves the right to make changes to the specifications contained herein without further notice.

The copyright of the product specification and the final right of interpretation of the product belong to Sencoch.

# 1 Product Description

The GZP6869DC pressure sensor is a state-of-the-art MEMS pressure sensor designed for a wide applications in medical&health care,industry&automation, household appliances and consumer electronics with specific pressure range. It is composed of a silicon piezoresistive pressure sensing chip and a signal conditioning integrated circuit. The initial signal from the sensing chip is amplified, temperature compensated, calibrated and finally converted to a digital signal(I2C) that is corresponding to the applied pressure.

## 1.1 Features

- Multiple range from -100…0 to 40…500kPa
- Gauge pressure type
- SOP8 package
- Power supply voltage: 2.7V to 5.5V
- IIC Interface
- Suitable for non-corrosive gases or liquids

## 1.2 Applications

- Respirators, oxygen generators, monitors, nebulizers and other equipment in the medical field
- Negative pressure measurement, pressure instruments, pneumatic components and other fields
- Massagers, massage chairs, air mattresses and other sports and fitness equipment
- Vacuum packaging machine, vacuum mixer, vacuum wall breaker, vacuum fresh-keeping box, vacuum pump and other vacuum negative pressure fields
- Washing machines, beer machines, coffee machines, vacuum cleaners, water purifiers, water heaters and other home appliances

# 2 Function Description

This product is manufactured using advanced micro-electromechanical principles. Its core technologies are a MEMS pressure sensor chip based on the silicon piezoresistive effect and a high-performance signal conditioning AISC chip. The silicon micro-piezoresistive MEMS pressure sensor chip forms a Wheatstone bridge through four strain-sensitive resistors. The output signal is amplified, temperature-compensated, and linearized by the ASIC chip. The linearization and temperature compensation of the transfer function are implemented by the digital processing circuit in the ASIC. Through the polynomial compensation algorithm and multi-point pressure calibration technology at multiple temperatures, high-precision pressure measurement is achieved over the entire operating temperature range.

## 2.1 Pin Definition

The pin configuration of the pressure sensor is shown in Figure 2.

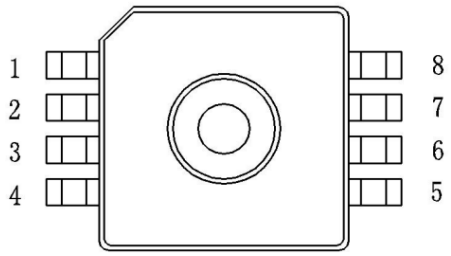



Fig.2 Pin configuration diagram

The corresponding relationship of the pressure sensor pins is shown in Table 1.

Tab.1 Pin Definition

| PIN No. | Description | Remark |
|---|---|---|
| 1 | SDA | Data line in I2C mode |
| 2 | NC | Floating pin |
| 3 | VDD | Power input positive |
| 4 | NC | Floating pin |
| 5 | GND | Power input negative |
| 6 | NC | Floating pin |
| 7 | SCL | Clock signal |
| 8 | NC | Floating pin |

## 2.2 Block Diagram

The functional block diagram of the pressure sensor is shown in Figure 1.

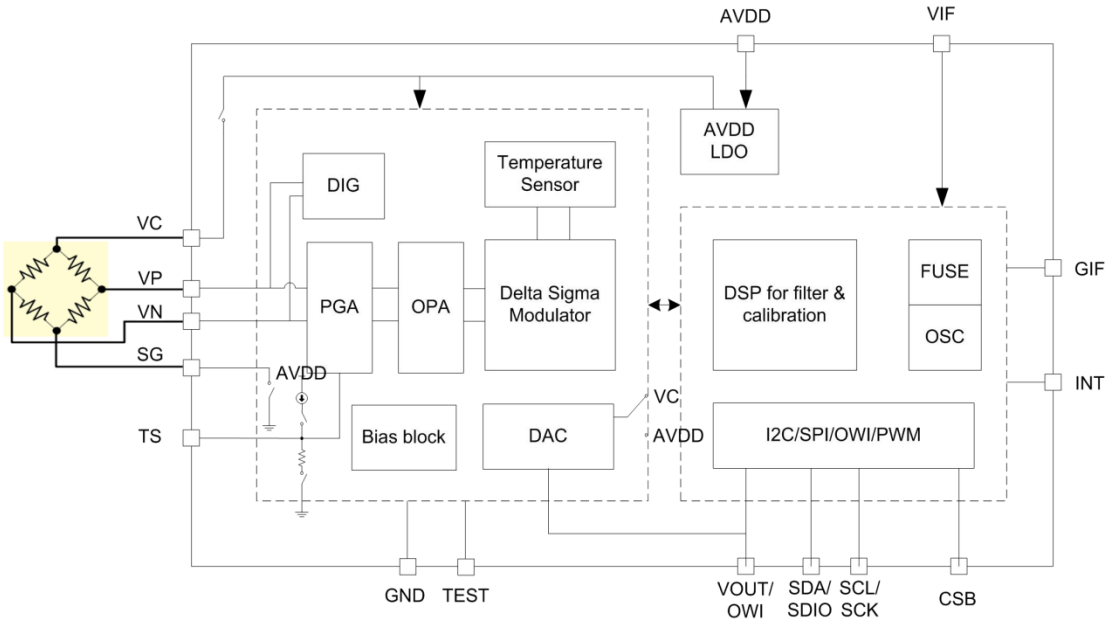



Fig.1 Block Diagram

## 2.3 Accuracy

GZP6869DC pressure sensor is affected by supply voltage, input pressure, ambient temperature, and aging. The value calculated using the transfer function is the sensor's specified value, also known as the theoretical value. The sensor's error is the difference between the actual output value and the specified output value at a specified input pressure.

### 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.

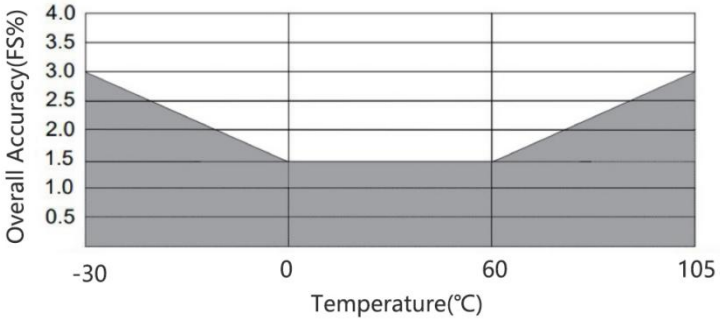



Fig.3 Overall Accuracy vs. temperature.

Tab.2 Overall Accuracy

| **Temperature (℃)** | **Overall Accuracy(Fs%)** |
|---|---|
| -30~105 | ±3.0% |
| 0~60 | ±1.5% |

* Different pressure ranges have different Overall Accuracy. Please consult customer service for more details.

# 3 Technical Indicators

The following indicators of the sensor are measured with (5±0.25)V DC and 25℃.

## 3.1 Maximum Ratings

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

Tab.3 The maximum rated parameters

| Parameter | Min. | Typical | Max. | Unit | Remark |
|---|---|---|---|---|---|
| Supply Voltage | -0.3 | | 6.5 | V | |
| ESD Protection | | ±2 | | kV | HBM |
| Overload Pressure | 4X (Range≤60kPa) | | | Rated | |
| | 2.5X (60kPa<Range<200kPa) | | | Rated | |
| | 1.5X (Range≥200kPa) | | | Rated | |
| Bursting Pressure | 5X (Range≤60kPa) | | | Rated | |
| | 3X (60kPa≤Range≤200kPa) | | | Rated | |
| | 2X (Range≥200kPa) | | | Rated | |
| Working | -30 | | 105 | ℃ | |
| 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

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

| Parameter | Value | Unit | Remark |
|---|---|---|---|
| Pressure Range | -100...0～5...500 | kPa | Customizable |
| Power Supply | 2.7~5.5 | V | |
| Standby Current | 100 | nA | |
| Accuracy | ±1 | %Span | |
| Pressure Resolution | 24 | Bits | |
| Built-in Temperature Accuracy | ±2 | ℃ | @0~60℃ |
| Temperature Resolution | 16 | Bits | LSB = (1/256) °C |
| Compensation Temperature | 0～60 | ℃ | Customizable |
| Pull-up Resistors | 2.2 | K ohm | |
| Clock Frequency | 400 | KHz | Max. |
| Response Time | 2.5 | ms | OSR_P=512X |

# 4 Application Circuit

The recommended application circuit of the sensor is shown in Figure 4.

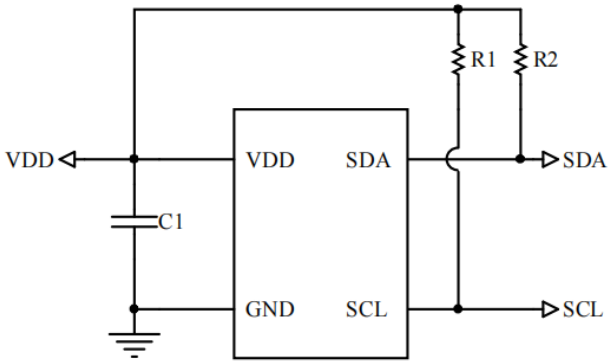



Fig.4 Recommended sensor application circuit diagram

**Notice :**

- The recommended value of C1 is 100nF, and the recommended values of R1 and R2 are 4.7k
- Please confirm the electrical definition before assembly
- Do not have any electrical connection to the NC pin, otherwise it may cause product failure.
- Provide anti-static protection during welding
- Overload voltage (6.5Vdc) may burn out the circuit chip
- This product has no reverse polarity protection, 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 0x58.

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.

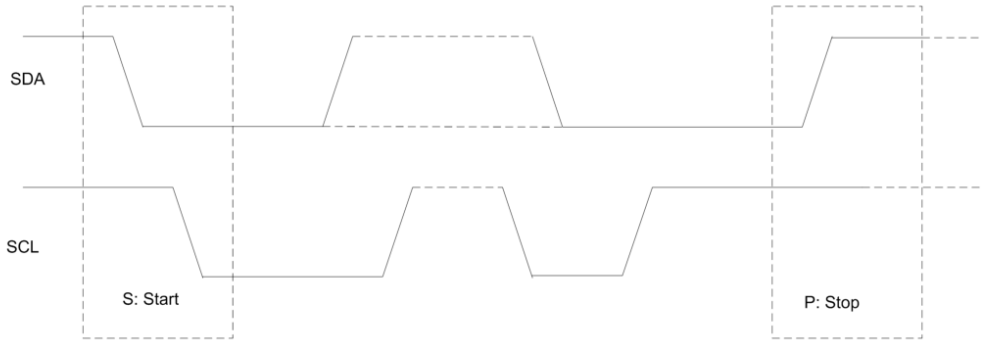



Fig.5. I²C Timing Diagram 1

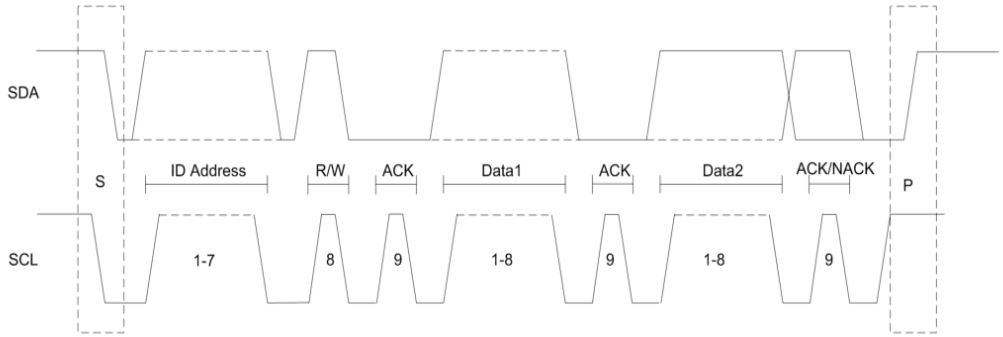



Fig.6 I²C Timing Diagram 2

# 6 Register Description

The register description is shown in Table 5.

Tab.5 Register description

| Add. | Description | R/W | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | ID | R | ID<7:0> | | | | | | | |
| 0x01 | Chip_Control | R/W | / | | data_ Ready | / | data_ou t | measurem ent_ctrl | Active<1:0> | |
| 0x02 | CFG_OSR | R/W | OSR_T<7:5> | | | OSR_P<4:2> | | | MODE[1:0] | |
| 0x03 | CFG_MEAS | R/W | / | | T_SB[5:3] | | | PT_R[2:0] | | |
| 0x04 | P_data | R | Data out<23:16> | | | | | | | |
| 0x05 | P_data | R | Data out<15:8> | | | | | | | |
| 0x06 | P_data | R | Data out<7:0> | | | | | | | |
| 0x07 | T_data | R | Temp out<15:8> | | | | | | | |
| 0x08 | T_data | R | Temp out<7:0> | | | | | | | |
| 0x24 | CFG_OPER | R/W | reserved<7:1> | | | | | | DAC_EN | |

**Reg0x00** $I^2C$ device address, the default address is 0x58.

**Reg0x01** (factory pre-configured)

Chip Control Register

active<1:0>: 00, the chip is powered off; 01, the chip is powered on;

measurement_ctrl: 0, pressure measurement; 1, temperature measurement;

data_out: 0, output calibration data; 1, output original data;

data_ready: 0, data conversion is not completed; 1, data conversion is completed.

**Reg0x02** (factory pre-configured)

MODE[1:0]: 00: Sleep mode, 01: Normal mode, 10: One shot mode

OSR_P[4:2]: (pressure oversampling):

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] (temperature oversampling):

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** (factory pre-configured)

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 (pressure/temperature measurement ratio in normal mode)

T_SB[5:3]: 000: 0ms, 001: 62.5ms, 010: 125ms, 011: 250ms, 100: 500ms, 101: 750ms, 110: 1000ms, 111: 2000ms (standby time setting in normal mode)

**Reg0x04-Reg0x06** Pressure Data Register

**Reg0x07-Reg0x08** Temperature Data Register

**Reg0x24** DAC_EN: 0: disable DAC, 1: enable DAC

# 7 Working Mode Description

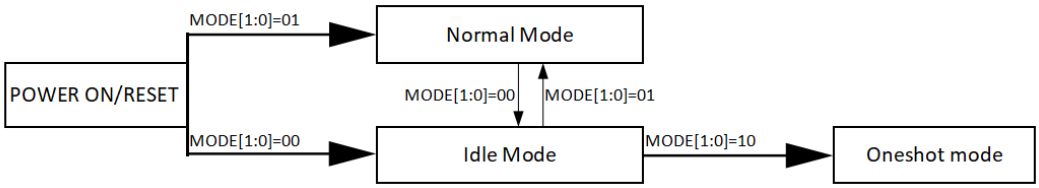



Fig.7 Working Mode

**Normal Mode:** When powered on, the sensor automatically enters Normal Mode. If switching from another mode to Normal Mode, it can be enabled by writing 01b to the MODE register (0x02[1:0]). The pressure and temperature sensor signals output measurement data cyclically at a predetermined frequency(Normally the standby time was pre-configured with 0mS).

**One Shot Mode:** It can be enabled by writing 10b to the MODE register (0x02[1:0]). The user can specify whether to measure the temperature or pressure signal by clearing or setting the measurement_ctrl bit (0x01[2]). After completing a single measurement, the sensor enters Idle Mode to await the next command.

**Idle Mode:** The sensor keep the low-consumption sleep situation till it is activated.

In normal mode, read 5 bytes continuously from 0x04 to 0x08 after power-up (ASIC will automatically refresh the data. The first 3 bytes are the pressure data, later 2 bytes are temperature data.

**Pressure Calculation**

Sum = (0x04 value * 2^16 + 0x05 value * 2^8 + 0x06 value),

If sum < 8388608, P=sum / 2^21*(PMAX - PMIN) (Unit: Pa)

If sum ≥8388608， P=(sum-2^24)/2^21*(PMAX - PMIN) (Unit: Pa)

※PMAX is the upper value of pressure range, PMIN is the lower value of pressure range.

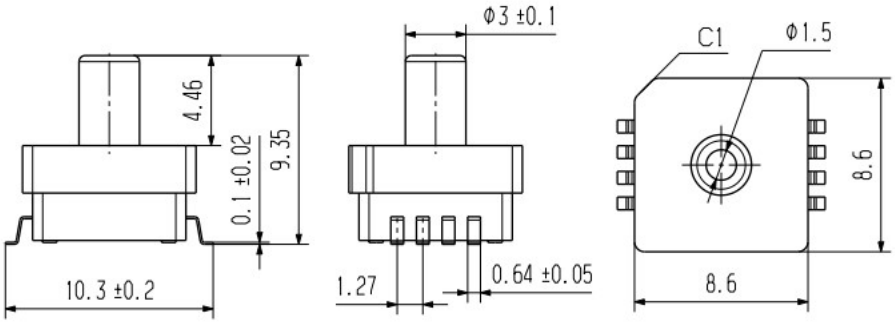
Tab.6 Range calibration parameters

| Pressure range | Calibration parameters | |
|---|---|---|
| | PMIN | PMAX |
| 0-3kPa | 0 | 3000 |
| 0-60kPa | 0 | 60000 |
| 0-350kPa | 0 | 350000 |

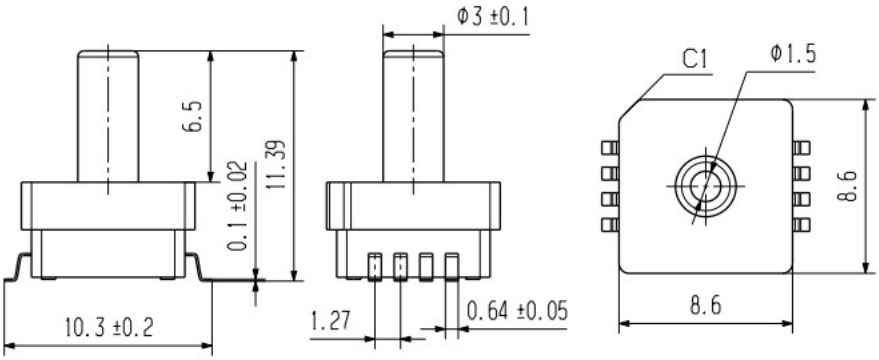
# 8 Structure Specification (unit:mm)

Refer to Figure 8 for the sensor's dimensions (error is ±0.1mm if not specified).

Type L0 （Air nozzle height: 4.46mm）




Type L1 （Air nozzle height: 6.50mm）



Type L1（Barbed nozzle height: 7.00mm）

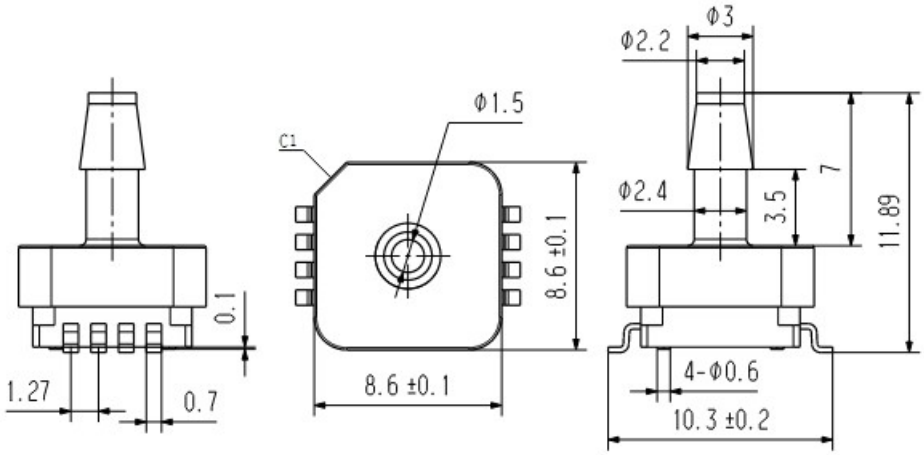



Fig.8 Product dimensions

The recommended pad dimensions are shown in Figure 9.

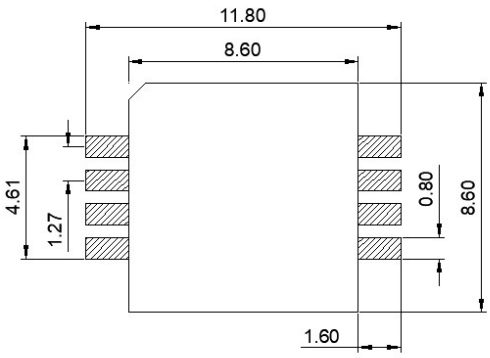



Fig.9 Recommended footprint

# 9 Order Guide

GZP 6869 D C - 101KPP F01 WX

Tab.7 Order rules

| | |
|---|---|
| GZP | Pressure Sensor Series |
| 6869 | Product Series |
| D | Output type A: Analog output D: IIC output |
| C | Communication protocol |
| 101KPP | Pressure range: 101 means the measured pressure value is 100 (including 0~100, -100~0, -100~100)<br>Pressure unit: KP: KPa MP: MPa PS: PSI BA: Bar<br>Pressure Type: P: positive pressure (e.g. 0~100) N: negative pressure (e.g. -100~0) W: negative pressure to positive pressure (e.g. -100~100)<br>Therefore, 100KPP represents the measured pressure from 0KPA to 100KPA |
| F 01 | Packaging Method: B01: Reel&Tape F01: Tube |
| WX | Company interior code |

# 10 Model Example

Tab.8 Model example

| **Pressure Range** | **Model** |
|---|---|
| 0 ~ 3kPa | GZP6869DC003KPPL0 F01WX |
| 0 ~ 60kPa | GZP6869DC060KPPL0 F01WX |
| 0 ~ 100kPa | GZP6869DC101KPPL1 F01WX |
| 0 ~ 200kPa | GZP6869DC201KPPL0 F01WX |
| 0 ~ 350kPa | GZP6869DC350KPPL0 F01WX |
| -100 ~ 0kPa | GZP6869DC101KPNL0 F01WX |
| -10 ~ 10kPa | GZP6869DC010KPWL1 F01WX |
| 0 ~ 200kPa | GZP6869DC201KPPL2 F01WX |

1. Above model example is for order information only, contact Sencoch for production and stock status.
2. For more customized ranges and special parameter part numbers, please consult Sencoch or agents.

# 11 Instruction 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

- Use a soldering iron with a head temperature between 260 and 300°C (30 W) and perform the work within 5 seconds.
- Please note that the output may change when soldering with a load applied to the terminals.
- Please keep the soldering iron tip clean.

( 2) Reflow soldering (SMD terminal type)

- The recommended wave oven temperature setting conditions are shown:

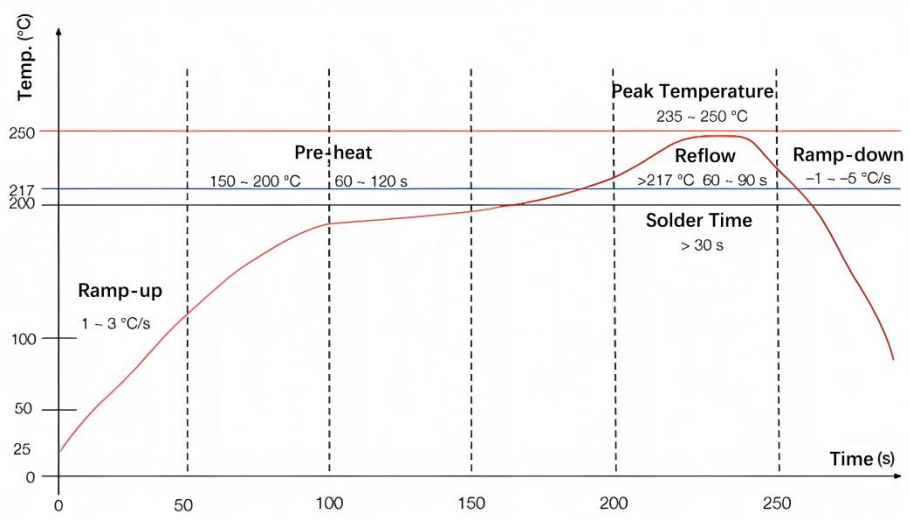



Fig.10 Remelting temperature setting conditions

( 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, be careful not to generate stress on the solder joint when cutting and bending the substrate.

( 5) Since the sensor terminals are exposed, contact with metal pieces or other objects may cause abnormal output. Be careful not to touch the terminals with metal pieces or your hands.

(6) When applying coating to prevent insulation degradation of the substrate after soldering, be careful not to allow chemicals to adhere to the sensor.

## 11.2 Cleaning Requirements

( 1) Since the product is open type, please 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 places 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 fluctuations in sensor output or damage.

( 3) Due to the structure of the pressure sensor chip, the output will fluctuate when it is exposed to light. Especially when applying pressure through a transparent cover, etc., please avoid light from reaching the sensor chip.

( 4) Normally packaged pressure sensors can be transported by ordinary transportation vehicles. Please note: The product must be protected from moisture, shock, 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 medium that can be used directly is non-corrosive gas or liquid. Other media, especially corrosive media or media containing foreign matter, may cause malfunction and damage. Therefore, please avoid using it in the above environment.

( 4) A pressure sensor chip is located inside the pressure inlet. Inserting a needle or other foreign object into the pressure inlet can damage the chip and clog the inlet, so please avoid such an operation.

( 5) Regarding the operating pressure, please use it within the rated pressure range. Using it outside the range may cause damage.

( 6) Since static electricity may cause damage, please be careful to ground charged objects on the table and operators when using it to allow the surrounding static electricity to discharge safely.

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

# 12 Packaging Information

Tube Packing:

Quantity per tube: 58 PCS

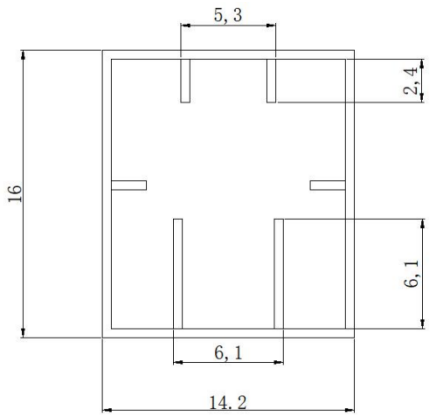
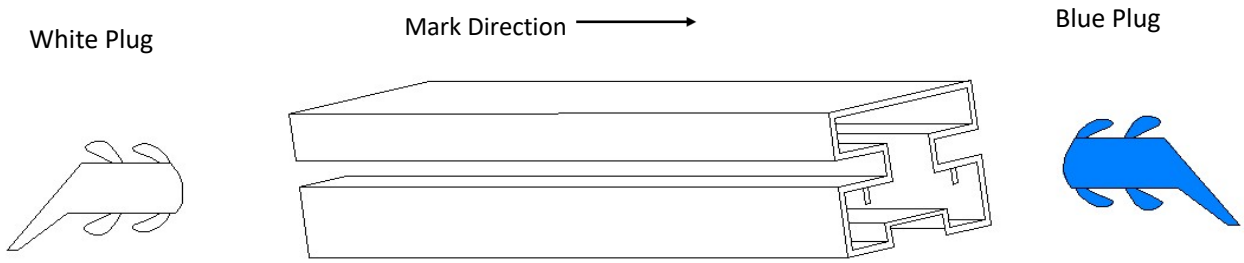
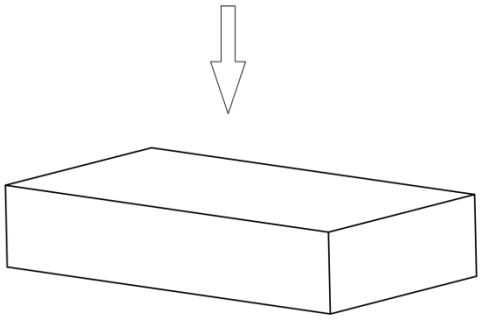



Fig.11 Section schematic diagram




（520mm x 14mm x 13mm，45PCS）



530mm x 145mm x 53mm，1800PCS

Fig.12 Outer Packing

## 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)**

```
#include "stm32f10x.h"

// Define SCL and SDA pins
#define SCL_PIN GPIO_Pin_6
#define SDA_PIN GPIO_Pin_7
#define I2C_GPIO_PORT GPIOB

// Pressure range
#define PMIN -100000.0
#define PMAX 200000.0
/*
Common range PIN PMAX
0 ~   4 kPa                    0.0          4000 .0
0 ~   5 kPa                    0.0          5000 .0
0   ~ 10kPa                    0 .0         10000 .0
0 20 kPa                       0.0          20000 .0
0   ~   40kPa                  0 .0         40000 .0
0 ~ 100kPa                     0.0          100000.0
0 ~ 200kPa                     0.0          200000.0
0 ~ 400kPa                     0.0          400000.0
0 ~ 700kPa                     0.0          700000.0
-1 ~ 1kPa                      -1000.0      1000.0
-5 ~ 5kPa                      -5000.0      5000.0
-20 ~ 20kPa                    -20000.0     20000.0
-100   ~   0 kPa               -100000 .0   0.0
-100   ~   100 kPa             -100000 .0   100000 .0
-100   ~ 1000 kPa              -100000 .0   1000000 .0
*/


// Delay function
void delay_us(uint32_t us)
{
uint32_t count = us * 7;
```

```
while (count--);
}

// I2C initialization function
void I2C_Init(void)
{
GPIO_InitTypeDef GPIO_InitStructure;
// Enable GPIOB clock
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
// Configure the SCL and SDA pins as open-drain outputs
GPIO_InitStructure.GPIO_Pin = SCL_PIN | SDA_PIN;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_OD;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(I2C_GPIO_PORT, &GPIO_InitStructure);
// Initial state, SCL and SDA are high
GPIO_SetBits(I2C_GPIO_PORT, SCL_PIN | SDA_PIN);
}

// void I2C_Start(void)
{
    GPIO_SetBits(I2C_GPIO_PORT, SDA_PIN);
    delay_us(5);
    GPIO_SetBits(I2C_GPIO_PORT, SCL_PIN);
    delay_us(5);
    GPIO_ResetBits(I2C_GPIO_PORT, SDA_PIN);
    delay_us(5);
    GPIO_ResetBits(I2C_GPIO_PORT, SCL_PIN);
    delay_us(5);
}

// void I2C_Stop(void)
{
    GPIO_ResetBits(I2C_GPIO_PORT, SDA_PIN);
    delay_us(5);
    GPIO_SetBits(I2C_GPIO_PORT, SCL_PIN);
```

```
    delay_us(5);
    GPIO_SetBits(I2C_GPIO_PORT, SDA_PIN);
    delay_us(5);
}

// void I2C_SendByte(uint8_t byte)
{
    for (uint8_t i = 0; i < 8; i++) {
        if (byte & 0x80) {
            GPIO_SetBits(I2C_GPIO_PORT, SDA_PIN);
        } else {
            GPIO_ResetBits(I2C_GPIO_PORT, SDA_PIN);
        }
        byte <<= 1;
        delay_us(5);
        GPIO_SetBits(I2C_GPIO_PORT, SCL_PIN);
        delay_us(5);
        GPIO_ResetBits(I2C_GPIO_PORT, SCL_PIN);
        delay_us(5);
    }
}

// I2C reads a byte and sends ACK or NACK
uint8_t I2C_ReadByte(uint8_t ack)
{
uint8_t byte = 0;
GPIO_SetBits(I2C_GPIO_PORT, SDA_PIN);
for (uint8_t i = 0; i < 8; i++) {
byte <<= 1;
GPIO_SetBits(I2C_GPIO_PORT, SCL_PIN);
delay_us(5);
if (GPIO_ReadInputDataBit(I2C_GPIO_PORT, SDA_PIN)) {
byte |= 0x01;
}
GPIO_ResetBits(I2C_GPIO_PORT, SCL_PIN);
```

```
delay_us(5);
}
if (ack) {
GPIO_ResetBits(I2C_GPIO_PORT, SDA_PIN); // Send ACK
} else {
        GPIO_SetBits(I2C_GPIO_PORT, SDA_PIN); // 发送 NACK
    }
    delay_us(5);
    GPIO_SetBits(I2C_GPIO_PORT, SCL_PIN);
    delay_us(5);
    GPIO_ResetBits(I2C_GPIO_PORT, SCL_PIN);
    delay_us(5);
    return byte;
}

// uint8_t I2C_WaitAck(void)
{
    uint8_t ack = 0;
    GPIO_SetBits(I2C_GPIO_PORT, SDA_PIN);
    GPIO_SetBits(I2C_GPIO_PORT, SCL_PIN);
    delay_us(5);
    if (GPIO_ReadInputDataBit(I2C_GPIO_PORT, SDA_PIN)) {
        ack = 1;
    } else {
        ack = 0;
}
GPIO_ResetBits(I2C_GPIO_PORT, SCL_PIN);
delay_us(5);
return ack;
}

// Read multiple bytes from the specified slave address and register address
void I2C_ReadBytes(uint8_t slave_addr, uint8_t reg_addr, uint8_t *data, uint8_t len)
{
I2C_Start();
```

```
//Send slave address, write operation
I2C_SendByte(slave_addr << 1);
I2C_WaitAck();
//Send register address
I2C_SendByte(reg_addr);
I2C_WaitAck();
I2C_Start();
//Send slave address, read operation
I2C_SendByte((slave_addr << 1) | 0x01);
I2C_WaitAck();
for (uint8_t i = 0; i < len; i++) {
if (i == len - 1) {
// Last byte, send NACK
data[i] = I2C_ReadByte(0);
} else {
// Send ACK
data[i] = I2C_ReadByte(1);
}
}
I2C_Stop();
}


int main(void)
{
uint8_t received_data[5];
uint8_t temp_coeff_data[2];
uint32_t pressure_data;
    uint16_t temperature_data;
    float shiftN;
int EOFFout;
float actual_pressure,actual_temperature;
I2C_Init();
// Slave address 0x58, register address 0x04, read 5 bytes continuously
I2C_ReadBytes(0x58, 0x04, received_data, 5);
```

```
// Combined pressure data
pressure_data = ((uint32_t)received_data[0] << 16) | ((uint32_t)received_data[1] << 8) | (uint32_t)received_data[2];
// Combined temperature data
temperature_data = ((uint16_t)received_data[3] << 8) | (uint16_t)received_data[4];
// Read two bytes from register 0x20 as temperature coefficient
I2C_ReadBytes(0x58, 0x20, temp_coeff_data, 2);
// Calculate shiftN
shiftN = (float)temp_coeff_data[1] / 10.0;
// Set EOFFout according to temperature coefficient [0]
    switch (temp_coeff_data[0])
    {
        case 0x0C:
            EOFFout = 4096;
            break;
        case 0x8C:
            EOFFout = -4096;
            break;
        case 0x0D:
            EOFFout = 8192;
            break;
        case 0x8D:
            EOFFout = -8192;
            break;
        case 0x0E:
            EOFFout = 16384;
            break;
        case 0x8E:
            EOFFout = -16384;
            break;
        default:
// You can add default processing, temporarily set it to 0 here
EOFFout = 0;
break;
}
```

```
// Process pressure data
if (pressure_data > 8388608)
        {
pressure_data -= 16777216;
}
// Process temperature data
if (temperature_data > 32768)
        {
temperature_data -= 65536;
}
// Calculate actual pressure
actual_pressure = ((float)pressure_data / (1 << 21)) * (PMAX -PMIN); // Unit is Pa
// Calculate actual temperature
actual_temperature = ((float)(temperature_data - EOFFout) / (1 << (int)shiftN)) + 25; //Unit is ℃
while (1)
        {
// Code to process received pressure, temperature, and temperature coefficient data can be added here
// For example, print data
// Note: To use printf you need to configure the serial port first
// printf("Pressure: %f, Temperature: %f,\n",actual_pressure,actual_temperature);
}
}
```